

# **Demand Access Processor Prototype Design Document (DAPPDD) Phase 1**

**February 25, 1999**



National Aeronautics and  
Space Administration

Goddard Space Flight Center  
Greenbelt, Maryland

**Demand Access Processor Prototype Design Document  
(DAPPDD) Phase 1**

February 25, 1999

**Initiated by:** \_\_\_\_\_

TBD  
Demand Access Project Manager  
Code 450

**Approved by:** \_\_\_\_\_

TBD  
Demand Access Project Manager  
Code 450

## **Preface**

This document was produced for the MO&DSD Networks Division, Code 450, and will be controlled by the Demand Access Program Manager.

Any questions or comments regarding this document should be addressed to:

Demand Access Project Manager  
Code 450  
Goddard Space Flight Center  
Greenbelt, Maryland 20771

<b>1. INTRODUCTION .....</b>	<b>9</b>
1.1   Background and Scope .....	9
1.2   Document Organization .....	13
1.3   Applicable Documents .....	14
<b>2. DAPP PHASE 1 PLAN.....</b>	<b>15</b>
2.1   High-Level Demonstration Scenario Script.....	16
2.2   DAP System Capabilities.....	17
2.3   Function Implementation Limitations.....	17
2.4   Phase 1 Detailed Design Task Organization.....	21
2.4.1   DAP Prototype System and Emulator Architecture Detailed Design Development (Task 1) 21	
2.4.2   DA PT Detailed Design Development (Task 2).....	21
2.4.3   Process A and DAB Emulator Applications Detailed Design Development (Task 3) .....	21
2.4.4   Process B DAP, NCC Emulator, and FDF Emulator Design Development Applications (Task 4)   24	
2.4.5   Processes C and D DAP Application and TGBFS Emulator Application (Task 5) .....	24
2.5   Phase 1 Hardware Configuration.....	24
<b>3. DEMAND ACCESS PLANNING TOOL DETAILED DESIGN .....</b>	<b>27</b>
3.1   Demand Access Planning Tool GUIs.....	27
3.1.1   Planning Tool Main Screen .....	28
3.1.2   Available Time Request Message Form .....	29
3.1.3   Schedule Add Request Message Form.....	30
3.1.4   Schedule Delete Request Message Form .....	31
3.1.5   DAP Report Form.....	32
3.1.6   Available Time Report Form .....	35
3.2   Demand Access Planning Tool PDL.....	37
3.2.1   PT_Main_GUI.....	38
3.2.2   DAPThread .....	39
3.2.3   PTNetwork.....	39
3.2.4   ATR-process .....	40
3.2.5   SAT-process .....	40
3.2.6   SDR-process .....	40
3.2.7   DAPMsg_process .....	41
3.3   Message Formats for the DAP to Planning Tool Interface .....	42
3.3.1   Available Time Request Message (From PT to DAP) (Msg 101) .....	43
3.3.2   Schedule Add Request Message ( From PT to DAP) (Msg 102).....	44
3.3.3   Schedule Delete Request Message ( From PT to DAP) (Msg 103) .....	45

3.3.4	Available Time Message ( From DAP to PT) (Msg 111) .....	46
3.3.5	Schedule Result Message ( From DAP to PT) (Msg112) .....	47
3.3.6	Schedule Deletion Notification Message ( From DAP to PT) (Msg 113).....	48
3.3.7	Forward Service Data Message ( From DAP to PT) (Msg 114).....	49
3.3.8	Return Service Data Message Format ( From DAP to PT) (Msg 115) .....	50

## 4. DAP PROCESS A SERVICE MANAGER DETAILED DESIGN.....51

<b>4.1</b>	<b>PDL Procedures for Process A .....</b>	<b>56</b>
4.1.1	Begin Process A.....	56
4.1.2	Procedure MAF Delete Arrival.....	56
4.1.3	Procedure Assess Message.....	57
4.1.4	Procedure MAF Schedule Delete Request.....	58
4.1.5	Procedure Verify MAF Service Exists.....	58
4.1.6	Procedure DAR Schedule Delete Request .....	59
4.1.7	Procedure Verify DAR Service Exists .....	59
4.1.8	Procedure Determine Directories.....	60
4.1.9	Procedure Delete TGBFS Messages .....	60
4.1.10	Procedure MAF service request.....	61
4.1.11	Procedure Implement MAF Service Request.....	61
4.1.12	Procedure DAR return request.....	62
4.1.13	Procedure Build TGBFS Messages .....	62
4.1.14	Procedure Load DARDB .....	63
4.1.15	Procedure Initialize DARDB Structure.....	64
4.1.16	Procedure Determine Required TGBFS Msgs.....	64
4.1.17	Process Build Msg Buffer .....	65
4.1.18	Procedure Insert Msgs into UserID File.....	66
4.1.19	Procedure MAF Planning Information.....	67
4.1.20	Procedure DAR Planning Information.....	67
4.1.21	Procedure Generate TDRS-USAT Visibility Schedule.....	68
4.1.22	Procedure Generate MAF User Availability Schedule .....	69
4.1.23	Procedure Generate DAR User Availability Schedule.....	70
4.1.24	Procedure Calculate TDRS-USAT Visibility .....	71
4.1.25	Procedure Update DAR User Schedule .....	72
4.1.26	Procedure Add User Request .....	73
4.1.27	Procedure Conflict Check.....	74
4.1.28	Procedure Delete User Request .....	74
4.1.29	Procedure Check MAF Add User Request .....	74
4.1.30	Procedure Build Message 111 .....	75
4.1.31	Procedure Build Message 112 .....	75
4.1.32	Procedure Build Message 113 .....	76
4.1.33	Procedure Build Message 114 .....	76
4.1.34	Procedure Build Message 115 .....	76
4.1.35	Procedure Build Message 200 .....	76
4.1.36	Procedure Build Message 300 .....	77
4.1.37	Procedure Build Message 400 .....	77
4.1.38	Procedure Build Message 401 .....	77
4.1.39	Procedure Build Message 500 .....	78
4.1.40	Procedure Build Message 501 .....	78
4.1.41	Procedure Availability Intervals .....	79
<b>4.2</b>	<b>Process A Message Formats.....</b>	<b>81</b>
4.2.1	Process A Output Messages.....	82
4.2.2	Process A Input Messages.....	82

4.2.3	Process A Data Outputs .....	83
4.2.4	Process A Data Inputs.....	84
<b>4.3</b>	<b>Process A, B, and C Shared Files and Process A Internal Files.....</b>	<b>85</b>
4.3.1	MAF TUT FILE RECORD FIELDS .....	85
4.3.2	TDRS INTERP EPHemeris FILE RECORD FIELDS .....	85
4.3.3	TEMPORARY TUT FILE RECORD FIELDS .....	85
4.3.4	TEMPORARY AVAILABILITY FILE RECORD FIELDS .....	86
4.3.5	TEMPORARY DIRECTION COSINE FILE RECORD FIELDS .....	86
4.3.6	TEMPORARY DIRECTORY FILE RECORD FIELDS .....	86
4.3.7	TEMPORARY MESSAGE FILE RECORD FIELDS.....	86
4.3.8	TEMPORARY USER FILE RECORD FIELDS .....	87
4.3.9	TEMPORARY USERID FILE RECORD FIELDS.....	87
4.3.10	USAT INTERP EPH FILE RECORD FIELDS.....	87
4.3.11	MAF USER AVAILABILITY SCHEDULE FILE RECORD FIELDS .....	87
4.3.12	DAR USER AVAILABILITY SCHEDULE FILE RECORD FIELDS .....	88
4.3.13	MAF USER SCHEDULE FILE RECORD FIELDS .....	88
4.3.14	DAR USER SCHEDULE FILE RECORD FIELDS .....	88
4.3.15	VISIBILITY SCHEDULE FILE RECORD FIELDS .....	88
4.3.16	TGBFS Messages for Phase 1 Scenario.....	89

## 5. DAP PROCESS B DAB DATA BASE MANAGER DETAILED DESIGN .....90

<b>5.1</b>	<b>PDL Procedures for Process B.....</b>	<b>92</b>
5.1.1	Begin Process B .....	92
5.1.2	Procedure Assess Message.....	93
5.1.3	Procedure Assess Msg 200 .....	94
5.1.4	Procedure Modify TUT .....	94
5.1.5	Procedure Add MAF Service .....	95
5.1.6	Procedure Delete MAF Service .....	96
5.1.7	Procedure Create Entirely New TUT .....	97
5.1.8	Procedure Enlarge TUT at Start.....	97
5.1.9	Procedure Enlarge TUT at End.....	97
5.1.10	Procedure Get TUT .....	98
5.1.11	Procedure Get Ephem .....	98
5.1.12	Procedure Start or Stop Getting Ephem.....	98
5.1.13	Procedure Create Interp EPH File.....	99
<b>5.2</b>	<b>Process B Message Formats.....</b>	<b>100</b>
5.2.1	Process B Input Messages.....	100
5.2.2	Process B Data Inputs .....	100

## 6. DAP PROCESS C TGBFS DATA PREPARATION DETAILED DESIGN ..101

<b>6.1</b>	<b>PDL Procedures for Process C .....</b>	<b>104</b>
6.1.1	Begin Process C .....	104
6.1.2	Procedure Build_Block.....	105
6.1.3	Procedure Gather TGBFS Msg Data .....	105
6.1.4	Procedure Determine Directories.....	105
6.1.5	Procedure Extract Msgs .....	106
6.1.6	Procedure Get UserID File Name .....	106
6.1.7	Procedure Build Segments .....	106
6.1.8	Procedure Make Group Segment Time Header .....	107

6.1.9	Procedure Insert TGBFS Messages into Block.....	107
<b>6.2</b>	<b>Process C Message Formats.....</b>	<b>113</b>
6.2.1	Process C Input Messages.....	113
6.2.2	Process C Data Inputs .....	113
6.2.3	Process C Output Messages .....	113
6.2.4	Process C Data Outputs .....	114
<b>6.3</b>	<b>Timing Requirements Among DAP Processes.....</b>	<b>115</b>
<b>7. DAP PROCESS D TGBFS DATA DISTRIBUTION DETAILED DESIGN ..</b>		<b>119</b>
<b>7.1</b>	<b>PDL Procedures for Process D .....</b>	<b>121</b>
7.1.1	Begin Process D.....	121
7.1.2	Procedure Build TGBFS Messages .....	122
<b>7.2</b>	<b>Process D Message Formats.....</b>	<b>123</b>
7.2.1	Process D Input Messages .....	123
7.2.2	Process D Output Messages.....	123
7.2.3	Process D Data Inputs.....	123
7.2.4	Process D Data Outputs .....	123
<b>8. DEMAND ACCESS BUFFER (DAB) EMULATOR DETAILED DESIGN ...</b>		<b>124</b>
<b>8.1</b>	<b>PDL Procedures for DAB Emulator .....</b>	<b>124</b>
8.1.1	Process DAB Emulator .....	124
<b>9. FLIGHT DYNAMICS FACILITY (FDF) EMULATOR DETAILED DESIGN .</b>		<b>126</b>
<b>9.1</b>	<b>PDL Procedures for FDF Emulator .....</b>	<b>126</b>
9.1.1	Process FDF Emulator .....	126
9.1.2	Procedure Propagate State Vector .....	127
9.1.3	Procedure Attitude Parameters.....	127
<b>10. NETWORK COMMUNICATIONS CENTER (NCC) EMULATOR DETAILED DESIGN.....</b>		<b>129</b>
<b>10.1</b>	<b>PDL Procedures for NCC Emulator .....</b>	<b>129</b>
10.1.1	NCC Message Handler Process .....	129
10.1.2	NCC Emulator Operator Process .....	130
10.1.3	Procedure Translate MAF TUT .....	131
<b>11. TGBFS EMULATOR DETAILED DESIGN .....</b>		<b>133</b>
<b>11.1</b>	<b>PDL Procedures for TGBFS Emulator.....</b>	<b>133</b>
11.1.1	Begin Process TGBFS Emulator.....	134
<b>11.2</b>	<b>TGBFS Emulator Message Formats .....</b>	<b>135</b>
11.2.1	TGBFS Emulator Input Messages .....	135
11.2.2	TGBFS Emulator Output Messages.....	135

12. DAP PROTOTYPE INTERFACE MESSAGE NUMBERING SCHEME ...	136
13. ABBREVIATIONS AND ACRONYMS .....	137
14. DAP PDL PROCEDURE INDEX .....	138

## 1. Introduction

### 1.1 Background and Scope

The Demand Access Processor Prototype (DAPP) contains a subset of the Demand Access System (DAS) functions described in the following documents:

- Demand Access Project Plan
- Demand Access Description and Operations Concept Document (DASDOC),
- Demand Access System Requirements (DASRS), and
- Demand Access System Interface Control Document (DASICD).

The function definitions presented in these documents provide the basis of the DAPP. The DAPP consists of the Demand Access Planning Tool (PT), the Demand Access Processor (DAP), and the Demand Access Buffer (DAB). Unlike the DAS, the DAPP excludes the Third Generation Beam Forming System (TGBFS) and the Return Link Data Recovery System (RLDRS) as integral parts of the system. The TGBFS function is relegated to be an emulator that interfaces with the DAPP. The RLDRS function is not implemented at all in the current version of the DAPP. As such, the DAPP implements only the control functions of the DAS outside of the TGBFS and RLDRS. The DAPP is a test bed for use in the feasibility testing of key DAS functions outside of the TGBFS and RLDRS. When connected to the TGBFS and RLDRS, the DAPP and these two functions become the DAS.

DAPP requirements and design stem from the DASRS and DASICD. The requirements for the DAPP are outlined in the document entitled DAPP Test Bed Requirements. The preliminary design for the DAPP is outlined in the document entitled DAPP Preliminary Design. The DAPP requirements and preliminary design form the basis for the information presented in this detailed design document.

The hardware components of the DAPP are shown in Figure 1-1. The Flight Dynamics Facility (FDF) and Network Communications Center (NCC) Emulators have replaced their real Goddard Space Flight Center (GSFC) counter parts. A TGBFS Emulator accepts commands and direction cosine data generated by the DAP. Multiple PT platforms interface with the DAP indicating the platform independent nature of the underlying the JAVA programming language based PT software.

Since the DAP is a software intensive system, the major portions of this document present the software blue prints for the DAPP. The logic flows from the preliminary design and addresses the details as to how the requirements are to be implemented in software. The blue print is presented in the form of a Program Design Language (PDL). The detailed design indicates specifically how the system requirements have been partitioned among processing entities and hardware. The PDL presents the logic that is to be used in implementing the software design at the code level. While it resembles code, it is in most instances at a level of detail above the code and assumes nothing about the coding language in which the system will be implemented.

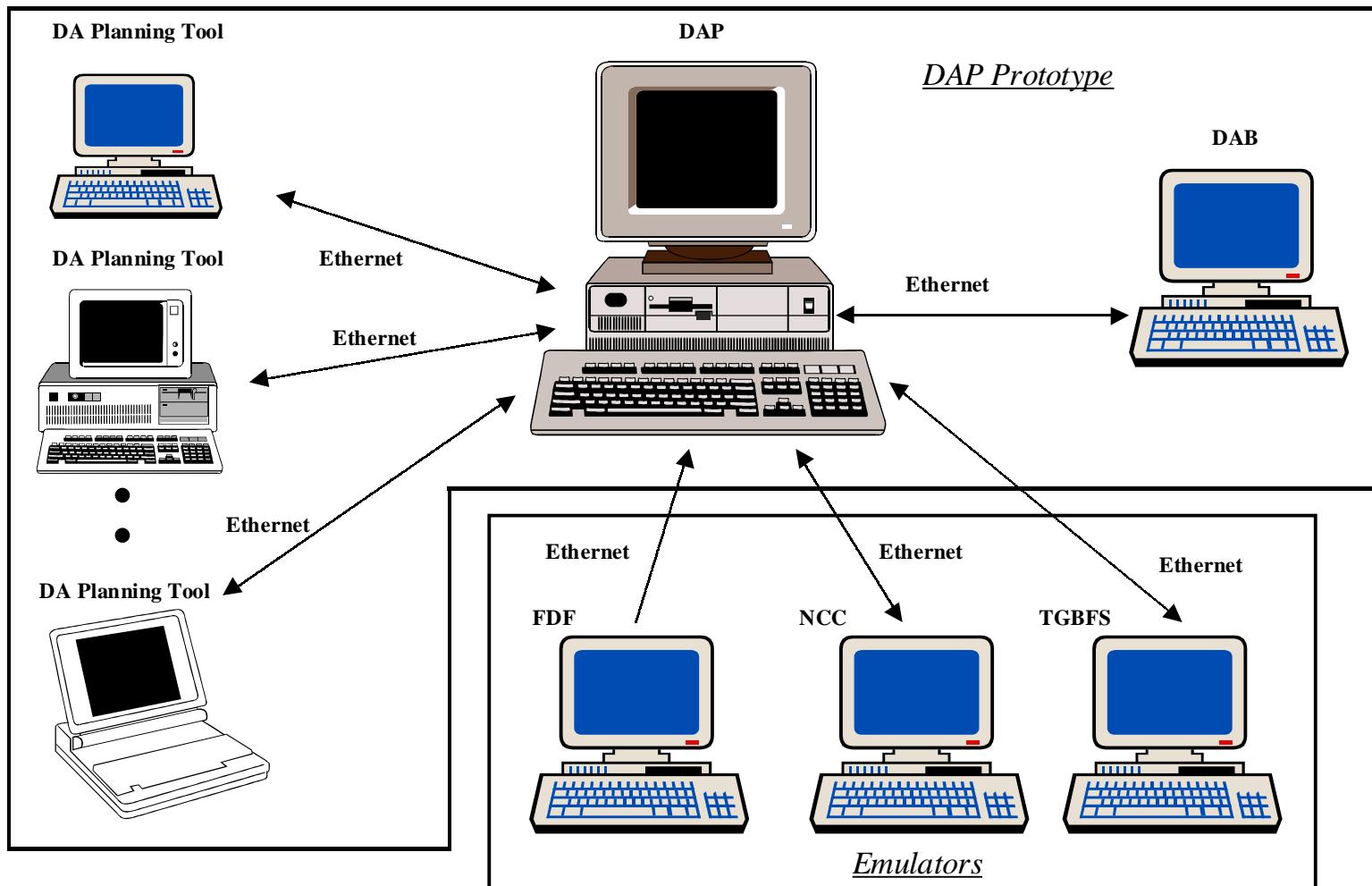


Figure 1-1: DAP Prototype Test Bed Hardware

Figure 1-2 shows the full software architecture of the DAP that has been established as part of the preliminary design process. The DAP is the major information processing center of DAS and the DAPP that provides link between the user friendly front end PT and the SN equipment required to implement the Multiple Access (MA) forward and return services. The DAP performs the following functions:

- Schedules MA forward and return services for DA users,
- Provides MA service planning information for DA users,
- Accepts TDRS Unused Time (TUT) schedules from the NCC,
- Accepts satellite state vectors and ephemerides from the FDF for propagation and interpolation, respectively,
- Uses TDRS and user orbit profiles for visibility evaluations of DA user provided information and service requests, and
- Generates the direction cosines based on orbit geometries to drive the TGBFS for real-time user return service support in advance of the service start time and stores them in a database for near real-time extraction and real-time distribution to the TGBFS.

Figure 1-2 shows the structure of the DAP software processes that meet the DAPP system requirements. It represents a DAPP that can interface with multiple SN users and three Space Ground Link Terminals (SGLTs) for worldwide Demand Access service implementation. This architecture supports a continued growth in processing requirements that is to be expected with a forecasted growing Demand Access user community. Phase 1 of the DAPP establishes foundation for this software architecture without implementing all of the hardware and software required to realize the full DAPP shown in Figure 1-2.

Phase 1 of the DAPP represents the first controlled installment in the construction of a fully operational DAPP. The basic difference between the Phase 1 DAPP and the full DAPP is the number of Space Network (SN) users that can be supported. The Phase 1 DAPP will be able to support one SN user and one Tracking and Data Relay Satellite (TDRS) for both forward and return Multiple Access (MA) services. Future phases of the DAPP will support multiple SN users of the MA services, multiple TDRSs, and multiple SGLTs. A detailed overview of Phase 1 of the DAPP is described in Section 2 of this document.

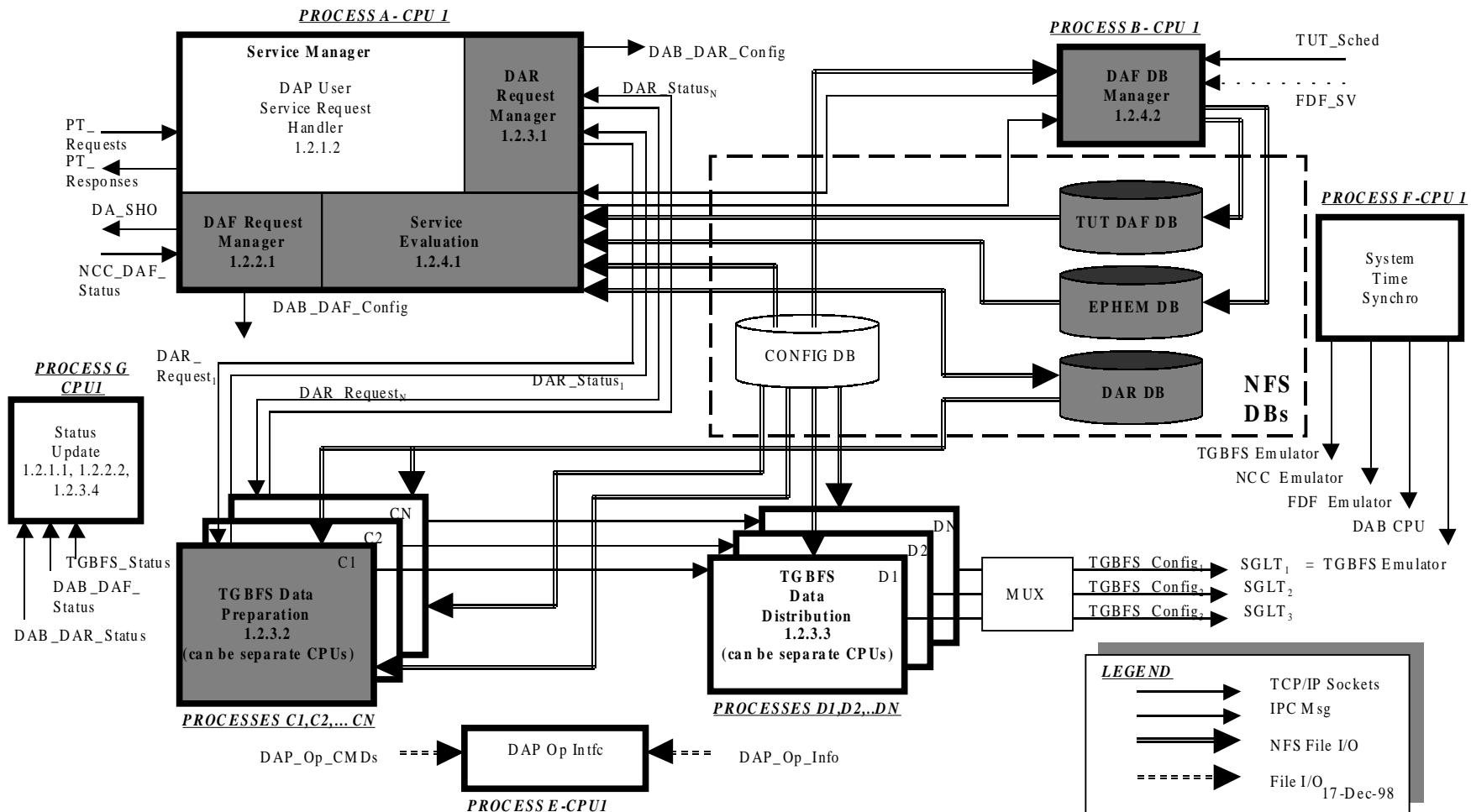


Figure 1-2: Final DAP Function Data Flow Diagrams to DAP Process Allocation

## 1.2 Document Organization

This document is divided into the 14 sections as summarized below.

- Introduction: Section 1 describes the scope and organization of this document as well as applicable documents
- DAP Phase 1 Plan: Section 2 of the document describes the objectives of Phase 1
- Demand Access Planning Tool Detailed Design: Section 3 contains the Graphical User Interfaces (GUIs), object oriented, event driven PDL, and message formats for the Planning Tool
- DAP Process A Service Manager Detailed Design: Section 4 contains the structured PDL and message formats for Process A of the DAP
- DAP Process B DAF Data Base Manager Detailed Design: Section 5 contains the structured PDL and message formats for Process B of the DAP
- DAP Process C TGBFS Data Preparation Detailed Design: Section 6 contains the structured PDL and message formats for Process C of the DAP
- DAP Process D TGBFS Data Distribution Detailed Design: Section 7 contains the structured PDL and message formats for Process D of the DAP
- Demand Access Buffer (DAB) Emulator Detailed Design: Section 8 contains the structured PDL for the DAB Emulator
- Flight Dynamics Facility (FDF) Emulator Detailed Design: Section 9 contains the structured PDL for the FDF Emulator
- Network Communications Center (NCC) Emulator Detailed Design: Section 10 contains the structured PDL for the NCC Emulator
- TGBFS Emulator Detailed Design: Section 11 contains the structured PDL for the TGBFS Emulator
- DAPP Interface Message Numbering Scheme: Section 12 contains the numbering assignments of groups of DAPP messages
- Acronyms and Abbreviations: Section 13 contains a list of all of the acronyms and abbreviations used in this document
- DAP PDL Procedure Index: Section 14 contains an index provided for quick reference to the specific DAP PDL procedures and various other items in the document.

### ***1.3 Applicable Documents***

The following documents were used as general references in the preparation of this document.

- Demand Access Project Plan, Version 1.1, February 1998.
- Demand Access System Description and Operations Concept (DASDOC), Version 1.3, February 1999.
- Specification for the Third Generation TDRSS MA Beamforming Subsystem Prototype Controller, Version 6.4, February 1999.
- Interface Control Document for the Third-Generation TDRSS MA Beamforming Subsystem, Revision 5.0, December 1998.
- Demand Access System Requirements Specification (DASRS), Version 1.1, February 1999.
- Demand Access System Interface Control Document (DASICD), Version 1.1, February 1999.
- Demand Access Processor Prototype Test Bed Requirements, Stanford Telecommunications, Inc., November 1998.
- Demand Access Processor Prototype Preliminary Design, Stanford Telecommunications, Inc., December 1998.
- Software Design Document (SDD) for the Demand Access (DA) Planning Tool, Revision 1.3, May 19,1998.

## 2. DAPP Phase 1 Plan

This section contains a first stage plan for the phased development of the DAPP starting with a demonstration baseline development system and evolving to greater and greater complexity towards a full DAPP. The evolutionary path from the baseline to the full DAPP is guided by the ideal capabilities of the operational DAPP as envisioned in the parent DAS specifications (see the DASRS and DASICD). The full DAPP is identified up front in order to provide a framework for augmenting the functional capabilities of each phase of DAPP development in an orderly progression towards a full DAPP. Figure 1-2 shows a full process based implementation of the DAP within the DAPP. The reason that it is a full DAP is that

- Its architecture represents a full DAP with the best current information on DAP expectations
- It takes into account the modular, multiple CPU expansion requirements of an operational DAP that needs continuous performance enhancements due to a growing DA user community in the operational environment
- It faces, up front, the system implementation problems associated with transitioning from a demonstration of simple DAP features to a fully operational DAP in the event that this requirement is levied at a future date
- It provides a skeletal framework onto which growing levels of DAS function capabilities can be added without severely impacting the system framework
- It satisfies the requirements and preliminary design specified in the DAPP documentation.

Note that demodulator management features for return service are not included since the RLDRS remains unspecified. This version of the operational system does not provide for forward and return command and data flows. It focuses on service control management aspects of DAP operations. Figure 1-2 shows the main processes of the Phase 1 of the DAP Prototype system. Phase 1 DAP functionality is tailored to meet the requirements of the Demonstration Scenario described below.

## 2.1 High-Level Demonstration Scenario Script

The following demonstration scenario exercises the DAP functions implemented in the Phase 1. The demonstration parallels the simple operations shown in Figures 3-3 and 3-4 of the DAS operations concept document (DASDOC) within the limitations of the emulators. Some minor nuances that are not described in this scenario may also be supported as part of a demonstration.

- NCC Emulator Operator issues TUT to DAP
- FDF Emulator Operator issues a TDRS and a USAT ephemeris to the DAP
- DAP User places a request for information for MAF opportunities based on TUT schedule and Visibility Schedule via the PT
- DAP reports back the MAF scheduling opportunities to the DA User
- DAP User selects a time segment out of the available time for MAF service request and submits request via the PT
- DAP schedules MAF forward service causing the NCC Emulator display to register the request for MAF service
- DAP provides DAB with MAF service buffer specifications for display at DAB
- DAP User places a request for information for MAF opportunities based on TUT schedule and Visibility Schedule to verify that TUT time has decreased
- DAP User places a request for DAR information based on USAT-TDRS visibility
- DAP reports back DAR scheduling opportunities to DAP User
- DAP User selects a time segment out of the available time for DAR service request and submits request specifying the dedicated TGBFS resource
- DAP schedules DAR service and implements it so that at the required time the direction cosines are sent to the TGBFS Emulator for a continuous scrolling display through the DAR scheduled service

## ***2.2 DAP System Capabilities***

The following capabilities refer to the system control implementation shown in Figure 1-2 and are tailored to meet the requirements of the Phase 1 Demonstration Scenario

- The processes A, B, C1, D1, E, and F are implemented on a single UNIX platform
- Multiple versions of the family of DAP processes C and D are not implemented
- Data Bases are local to the UNIX platform [no Network File System (NFS) implementation]
- TCP/IP Messages, and File I/O processes are used for the external and internal DAP process interface communications

## ***2.3 Function Implementation Limitations***

Table 2-1 shows the limitations to the functions being implemented in Phase 1. These limited function capabilities form the foundation for expansion in future phases.

Table 2-1 Phase 1 Function Implementation Limits

<b>Process</b>	<b>Function</b>	<b>Phase 1 Function Implementation Limits</b>
A	DAP User Service Request Handler 1.2.1.2	<ul style="list-style-type: none"> <li>• Receives 1 PT request for 1 USAT MAF planning information</li> <li>• Receives 1 PT request for 1 USAT MAF service request</li> <li>• Receives a request for 1 USAT dedicated DAR return request</li> <li>• Sends MAF planning information to PT</li> <li>• Sends MAF service request responses to PT</li> <li>• Sends DAR planning information to PT</li> <li>• Sends DAR service request responses to the PT</li> </ul>
A	MAF Request Manager 1.2.2.1	<ul style="list-style-type: none"> <li>• Forwards 1 MAF request "SHO" to NCC Emulator for display</li> <li>• Accepts MAF service implementation response from NCC Emulator</li> <li>• Updates the "working" TUT by removing scheduled time from the TUT</li> <li>• Provides the DAB with MAF command and data buffering specifications</li> </ul>
A	DAR Request Manager 1.2.3.1	<ul style="list-style-type: none"> <li>• Accepts request for 1 DAR request for dedicated TGBFS resource</li> <li>• Places DAR service specification information in DAR DB</li> <li>• Notifies the Process C that a DAR request is registered in the DAR DB</li> <li>• Formulates a service data buffering request</li> <li>• Provides the DAB with MAR command and data buffering specifications</li> </ul>

A	Service Evaluation Manager 1.2.4.1	<ul style="list-style-type: none"> <li>• For MAF planning information or service request, the following is done           <ol style="list-style-type: none"> <li>1. Extract TDRS and USAT ephemeris from the EPHEM DB</li> <li>2. Generates a Visibility Schedule for the TDRS-USAT for the request period</li> <li>3. Extracts TUT schedule from the TUT DB</li> <li>4. "Ands" the Visibility Schedule with the TUT Schedule to produce an Availability Schedule</li> <li>5. Provides the PT with the Availability Schedule if the request is for information</li> <li>6. "Ands" the Availability Schedule with the request interval for validation of resource availability</li> <li>7. If the resources are available, the request is honored and a notification for MAF service "SHO" is formulated</li> <li>8. The TUT DB schedule is updated to reflect the decrease in TUT due to the scheduled MAF service</li> <li>9. If the resources are not available, then a rejection notification is formulated</li> </ol> </li> <li>• For DAR request or service request, the following is done           <ol style="list-style-type: none"> <li>1. Extract TDRS and USAT ephemeris from the EPHEM DB</li> <li>2. Generates a Visibility Schedule for the TDRS-USAT for the request period</li> <li>3. "Ands" the Visibility Schedule with the Visibility Schedule to produce an Availability Schedule</li> <li>4. "Ands" the Availability Schedule with the request interval for validation of resource availability</li> <li>5. If visibility is verified, the request is honored and the Availability Schedule and TGBFS commands are placed in the DAR DB and the Process C1 is notified to implement the DAR service</li> <li>6. If visibility is achieved according to the request specification, then a notification of service failure is formulated</li> </ol> </li> </ul>
B	DAF DB Manager 1.2.4.2	<ul style="list-style-type: none"> <li>• Accepts a TUT schedule from the NCC Emulator, converts it to internal DAP representation, and places it in the TUT DB</li> <li>• Makes a "working" TUT for dynamic scheduling constraints</li> <li>• Accepts 1 TDRS and 1 USAT ephemeris from the FDF Emulator, converts it from FDF format to internal DAP representation, and stores it in the EPHEM DB data base</li> <li>• DB management is performed using built in command line UNIX file management capabilities</li> </ul>
C1	TGBFS Data Preparation 1.2.3.2	<ul style="list-style-type: none"> <li>• Accepts notification of DAR service request from Process A</li> <li>• Accesses the DAR DB for the DAR request Visibility Scheduling specifications</li> <li>• Place request on DAR scheduler for real-time implementation</li> <li>• Extract direction cosines from DAR DB and stores them in data blocks</li> <li>• Block send the direction sine and cosines to Process D1</li> </ul>
D1	TGBFS Distribution Data 1.2.3.3	<ul style="list-style-type: none"> <li>• Accept the direction cosine blocks for Process C1</li> <li>• Send the direction cosine message packets to the TGBFS Emulator at the real-time rate of once every 2 second</li> </ul>
E	DAP Operator Interface	<ul style="list-style-type: none"> <li>• Start and terminate DAP operations from the console via command line program execution (the standard console I/O support task that is used to create and kill the remaining processes)</li> </ul>
F	System Time Synchronization	<ul style="list-style-type: none"> <li>• Broadcasts "DAP time" to the Emulators</li> </ul>

G	Status Update 1.2.1, 1.2.2.2, 1.2.3.4	<ul style="list-style-type: none"><li>• Displays the emulated status of TGBFS and the DAB processors in a standard text console window at the DAP</li></ul>
---	--	---

## ***2.4 Phase 1 Detailed Design Task Organization***

The tasks below represent the work breakdown associated with the detailed design function implementation outlined in Table 2-1. It includes DAP software development as well as the Emulators software development work. Phase 1 software development will be targeted to the hardware platforms described in Section 2.5.

### **2.4.1 DAP Prototype System and Emulator Architecture Detailed Design Development (Task 1)**

- Design and develop UNIX process framework for DAP application programs with inter-process communications based on Figure 1-2 and Figure 2-1
- Design and develop communications software in the emulators to allow DAP and emulator applications programs to communicate over DAP external interfaces
- Design and develop the socket communication software on UNIX platforms to allow the applications software (in Tasks 2 through 5 below) to send and receive messages to and from the DAP
- This task covers system start up, initialization (configuration), and termination
- Design and develop the DAP Operator interface Task shown in process E
- Design and develop the DAP Status software shown in process G
- This software will be developed in the C Programming language (and UNIX shell scripts as needed).

### **2.4.2 DA PT Detailed Design Development (Task 2)**

- Design and develop a simple menu system of GUI input and output forms to support demonstration requirements
- Design and develop the software required to build messages containing data from input forms data destined for DAP Process A via the external interface
- Design and develop the software to required extract data from messages via the external interface from DAP Process A
- Design and develop socket software to communicate with Process A
- This task's software will be developed entirely in the JAVA Programming language for platform independence purposes.

### **2.4.3 Process A and DAB Emulator Applications Detailed Design Development (Task 3)**

- This software interfaces with the PT (Task 2), Process B (Task 4) and Process C (Task 5) applications software.
- Design request handler to assess the nature of the request and select the appropriate course of action
- Design the DAF schedule evaluation software that access the DAP database for TUT and ephemeris data managed by process B
- Design the DAR scheduled evaluation software to maintain DAR DB

- This task's software will be developed in the C Programming language.

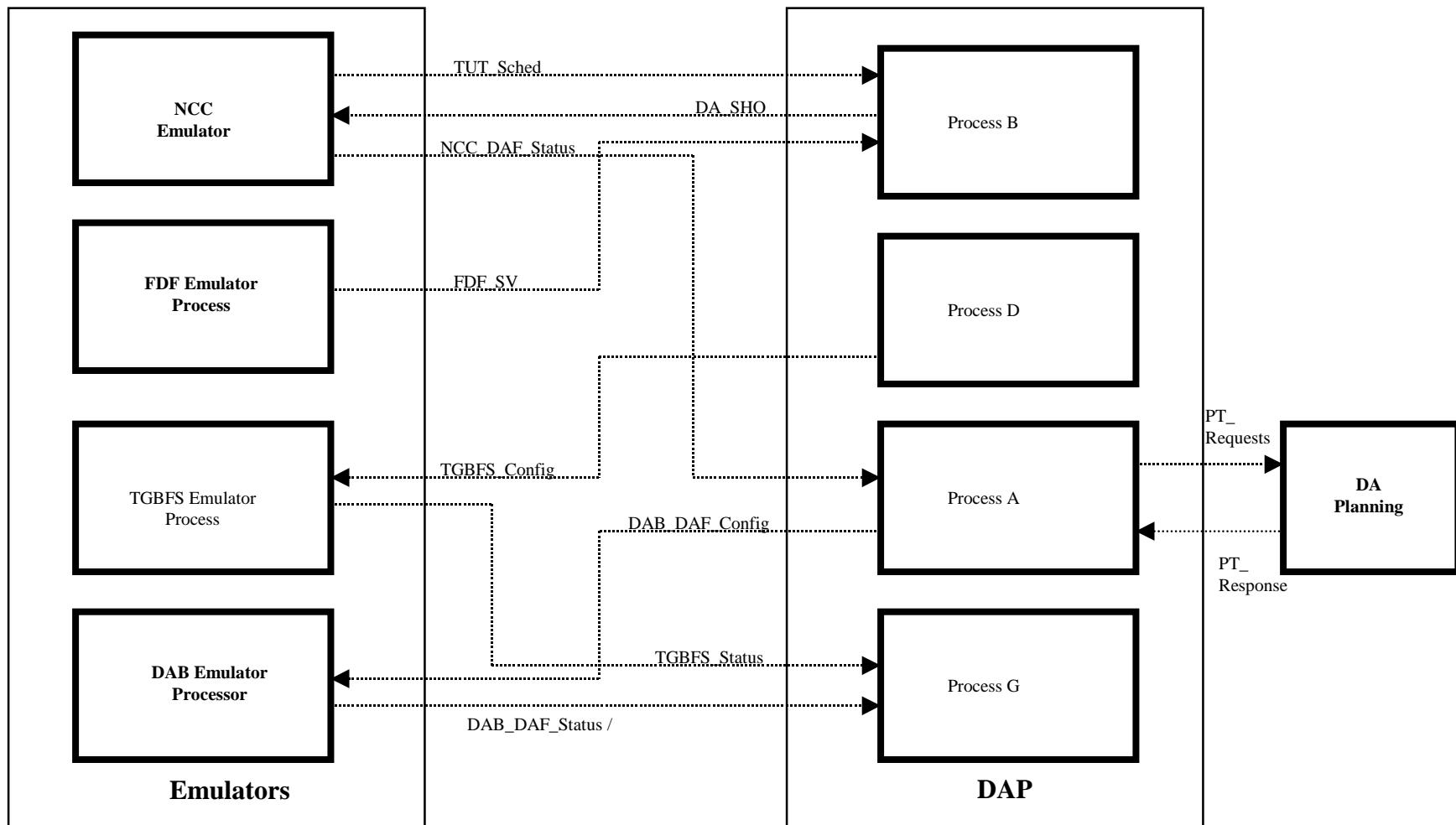


Figure 2-1: DAP TCP/IP External Interfaces

#### **2.4.4 Process B DAP, NCC Emulator, and FDF Emulator Design Development Applications (Task 4)**

- Design DAF database management software to extract NCC Emulator and FDF Emulator inputs
- Design DAF database management software such that **only** process B changes the database contents while other processes can only read the database
- This task's software will be developed entirely in the C Programming language.

#### **2.4.5 Processes C and D DAP Application and TGBFS Emulator Application (Task 5)**

- Design DAR scheduler software for process C
- Design TGBFS data preparation software for process C
- Design TGBFS interface software for process D
- Design the TGBFS Emulator process software
- This task's software will be developed in the C Programming language.

### ***2.5 Phase 1 Hardware Configuration***

Figure 2-2 shows the Phase 1 hardware configuration. The DAP software is situated on a SUN Workstation. The software for the four emulators (NCC, FDF, TGBFS, and DAB) is situated on a second SUN Workstation. Each emulator function is developed as a separate process that runs completely independent of the other emulators. Each emulator has it's own window that provides control and status display capabilities for that emulator. The communication paths between the emulator processes and the DAP processes are shown in Figure 2-1. The Ethernet networking system is used to implement the communication paths between the three platforms.

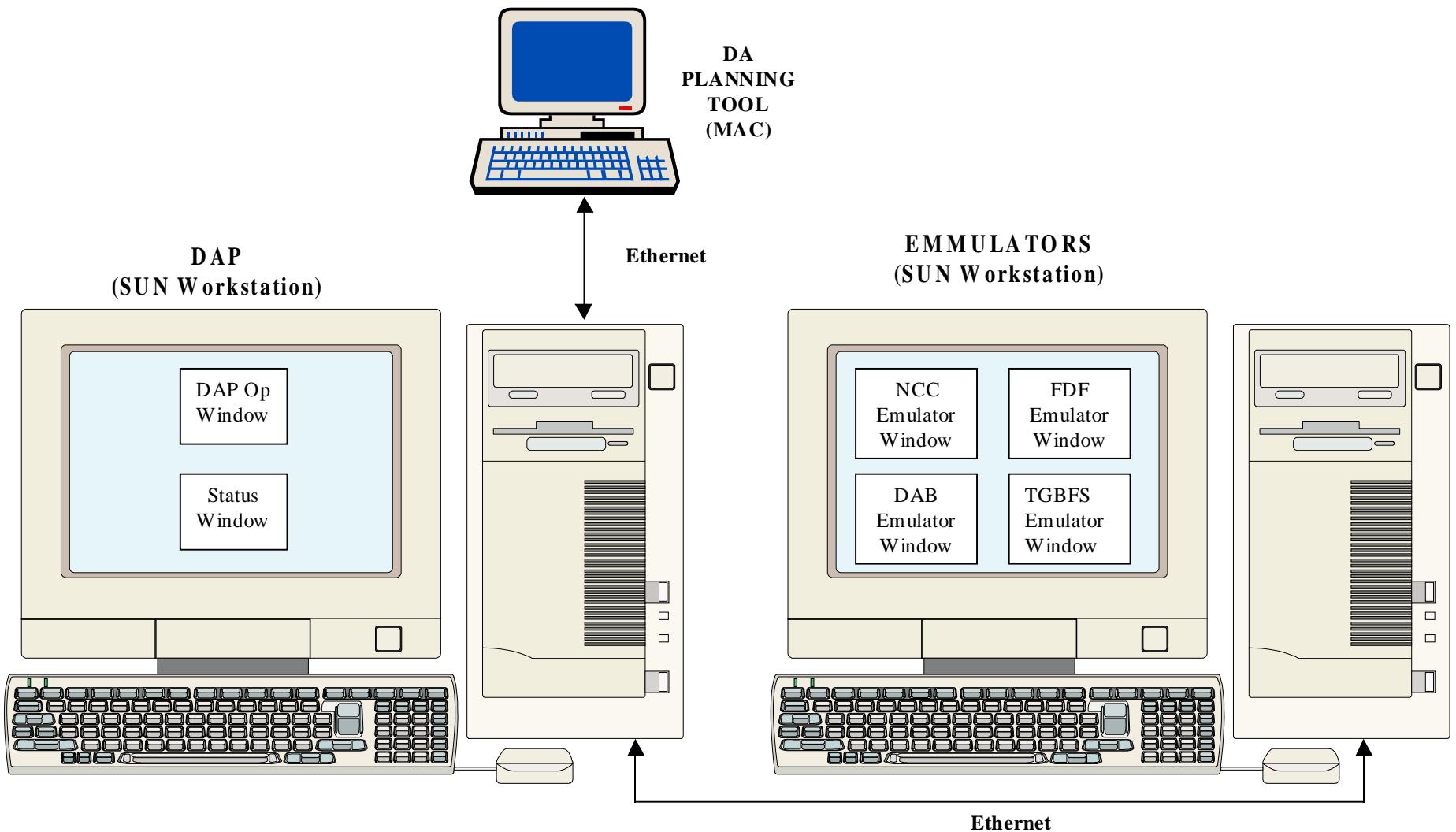


Figure 2-2: DAPP Phase 1 Hardware Architecture

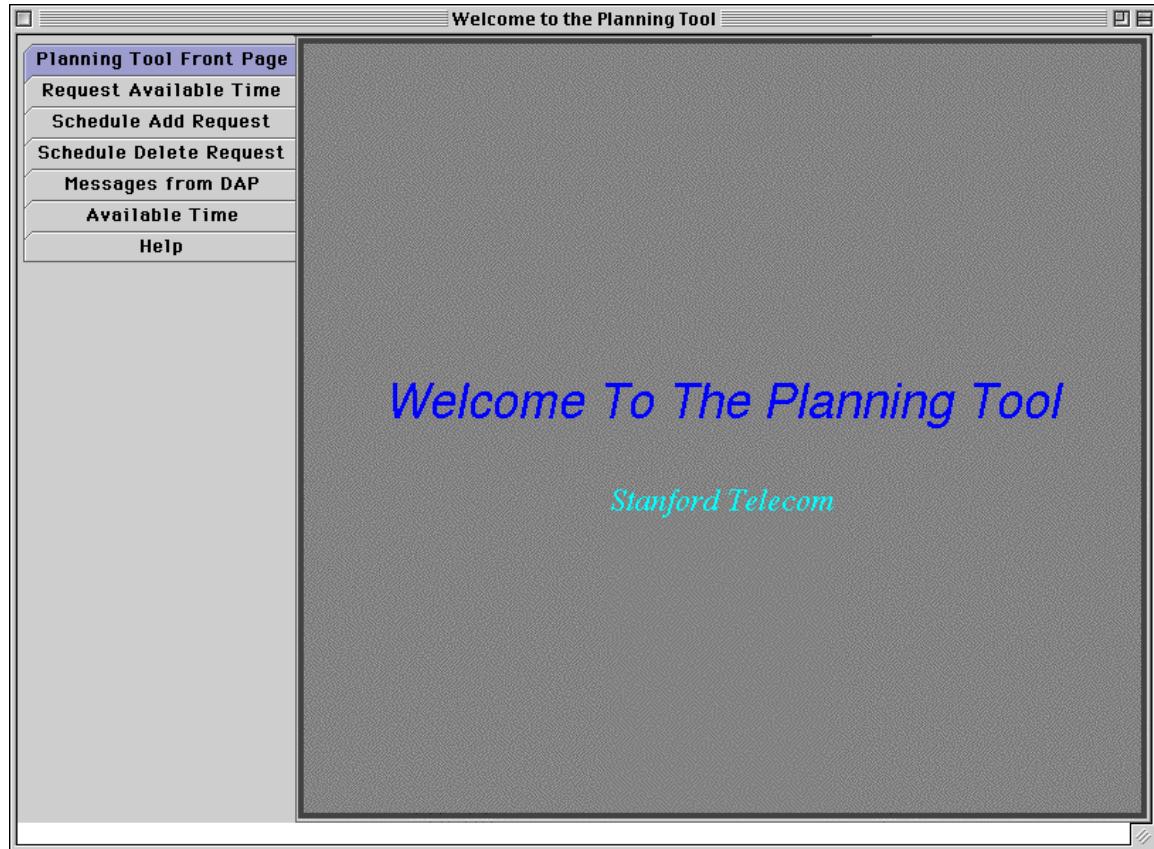
## 3. Demand Access Planning Tool Detailed Design

### 3.1 *Demand Access Planning Tool GUIs*

The following sections show samples of the range of forms and reports associated with user interactions. The Planning Tool (PT) is developed as a front-end Graphical User Interface (GUI) for communicating with the DAP. It will be implemented in the JAVA programming language, thus making it a platform independent application. The GUIs provide several forms that interactively accept user's input and communicate the information to the DAP. The results of the DAP processing the input information are communicated back to the user via additional screens or overlaid result screens.

### 3.1.1 Planning Tool Main Screen

Upon launching of the Planning Tool, the Main Planning Tool Screen shown in Figure 3-1 is displayed to the user. This screen displays the PT welcome page. The Left-hand side of the screen panel displays the tabs for all the possible user activity selections for the PT. When a tab is selected via clicking of the mouse button, the corresponding screen is displayed to the user.



**Figure 3-1: Planning Tool Main Screen**

### 3.1.2 Available Time Request Message Form

The Available Time Request Message Form shown in Figure 3-2 displays the GUI for PT users to construct and send an Available Time Request Message to DAP to query available time slots for a particular TDRS on a specific date for forward and return services. DAP will respond with an Available Time Message display (shown below in Figure 3-5) listing times available for up to 7 days from the requested date. After the “send” button is selected, a message is displayed at the bottom of the screen acknowledging success or failure of the scheduling action by the DAP.

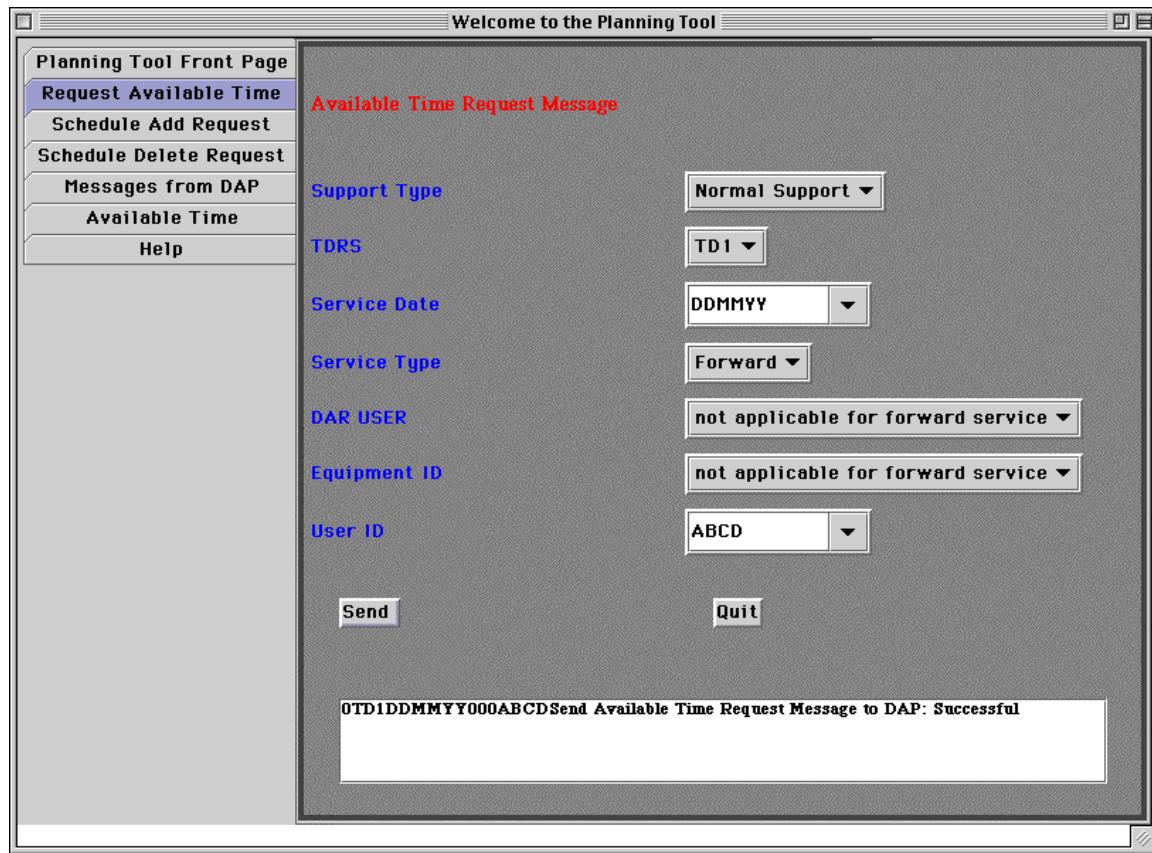


Figure 3-2: Available Time Request Message Form

### 3.1.3 Schedule Add Request Message Form

The Schedule Add Request Message Form shown in Figure 3-3 displays the GUI that allows PT users to construct and send a Schedule Add Request Message to DAP to add a specific schedule request.

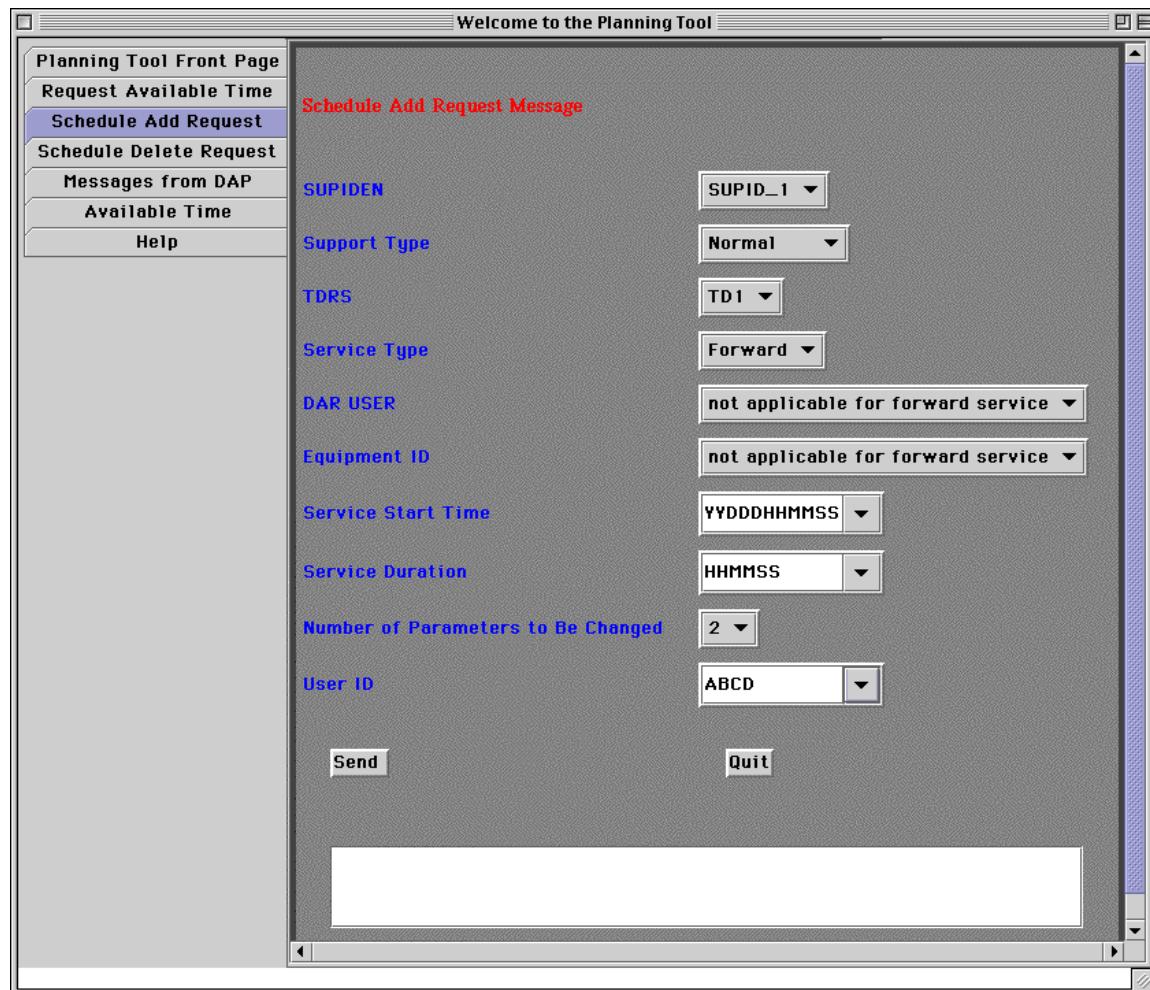


Figure 3-3: Schedule Add Request Message Form

### 3.1.4 Schedule Delete Request Message Form

The Schedule Delete Request Message Form shown in Figure 3-4 displays the GUI for PT users to construct and send a request to DAP to delete an already scheduled event. Message status is displayed in the text field that the message has been successfully sent to DAP.

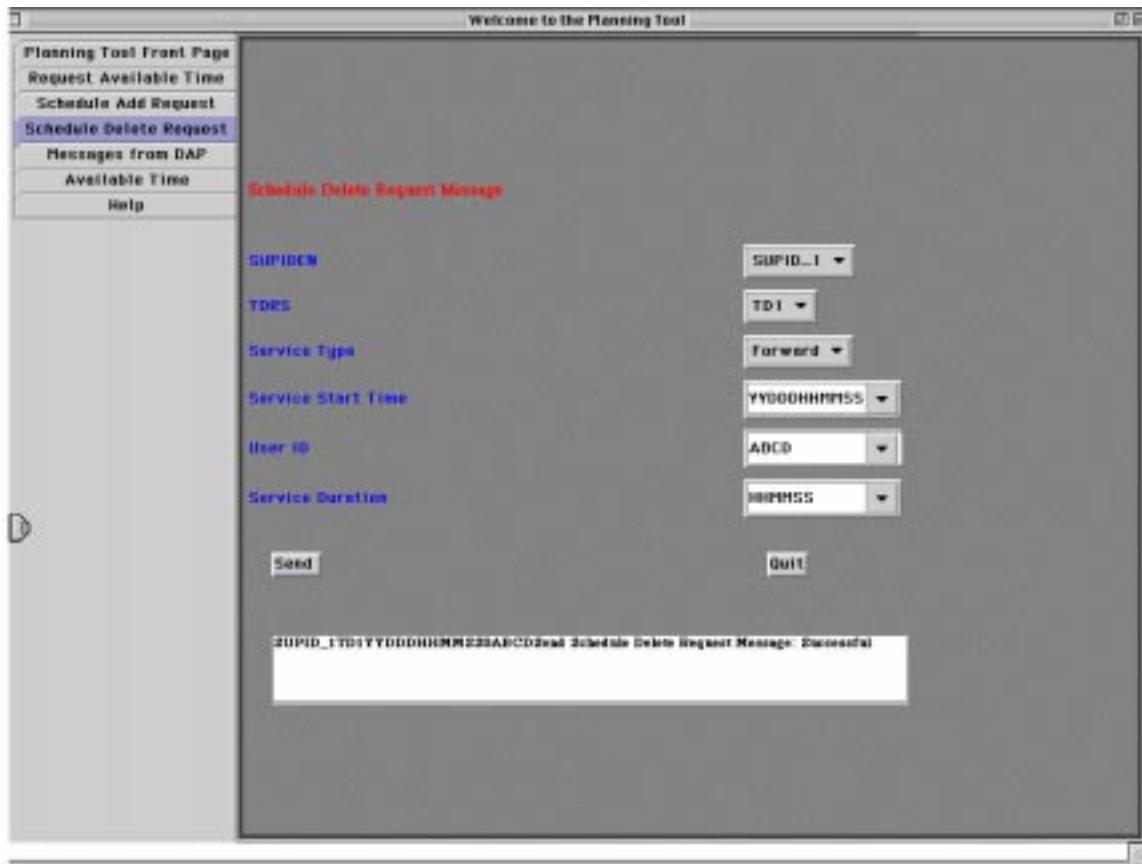
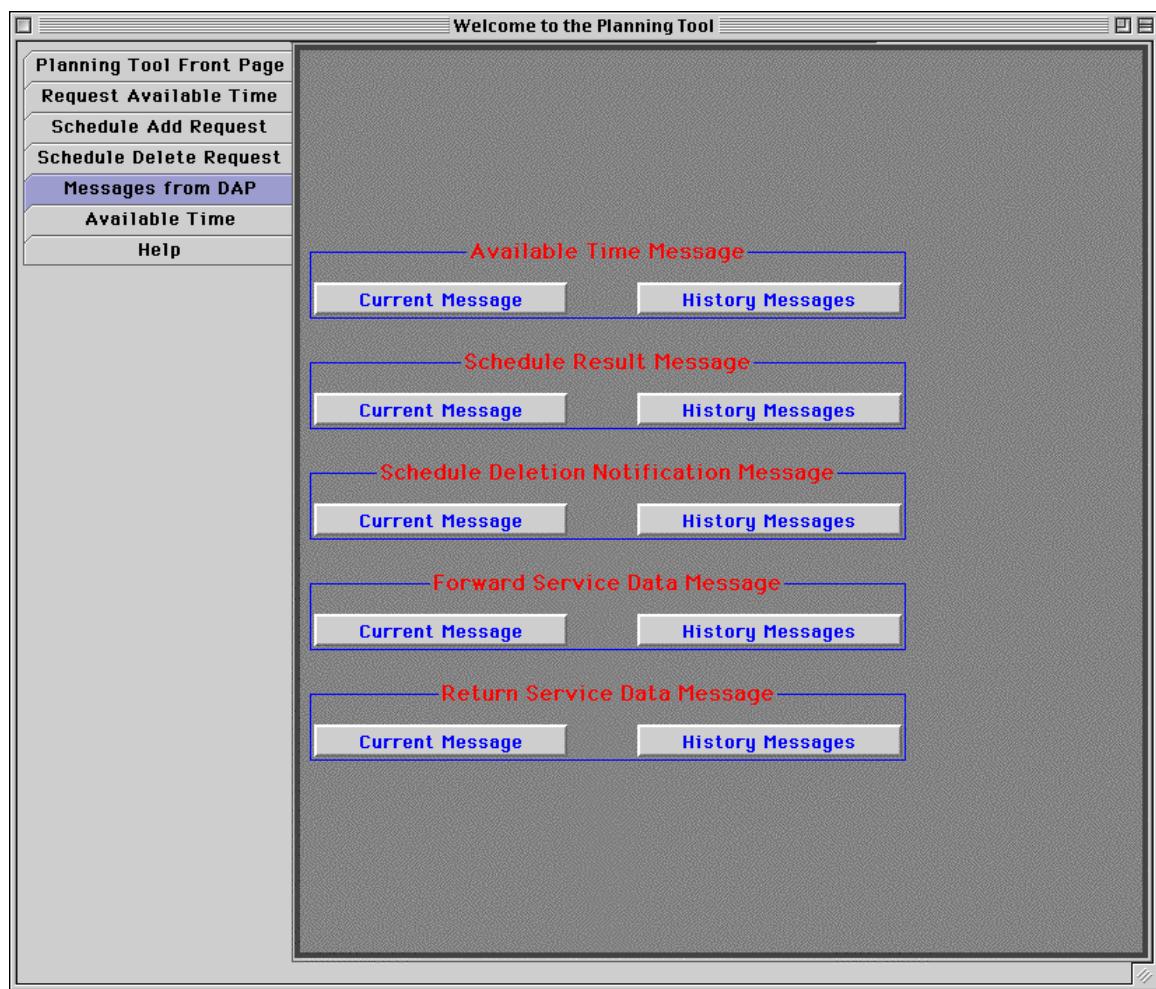


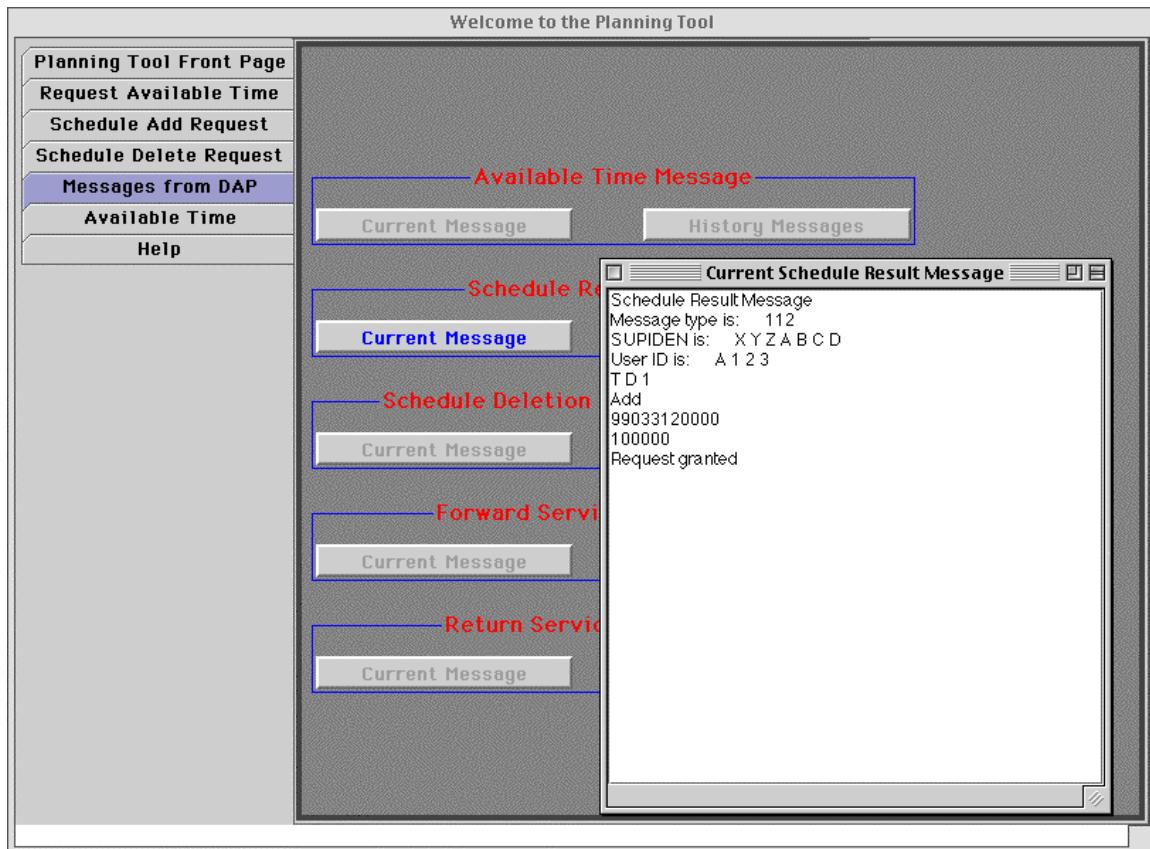
Figure 3-4: Schedule Delete Request Message Form

### 3.1.5 DAP Report Form

The DAP Report Form shown in Figure 3-5 displays all the information sent from DAP to PT. The PT user can make a selection to view the current message or history of messages for each type of message. When the "Current Message" button is selected, a new screen pops up with the display of the contents of the selected type message. If the "History Message" button is selected, the corresponding history of messages for the selected message type will be displayed on a new screen. Examples of these two output selections are shown in Figures 3-6 and 3-7, respectively.



**Figure 3-5: DAP Report Form**



**Figure 3-6: Schedule Result Message (Current Message) Report Form**

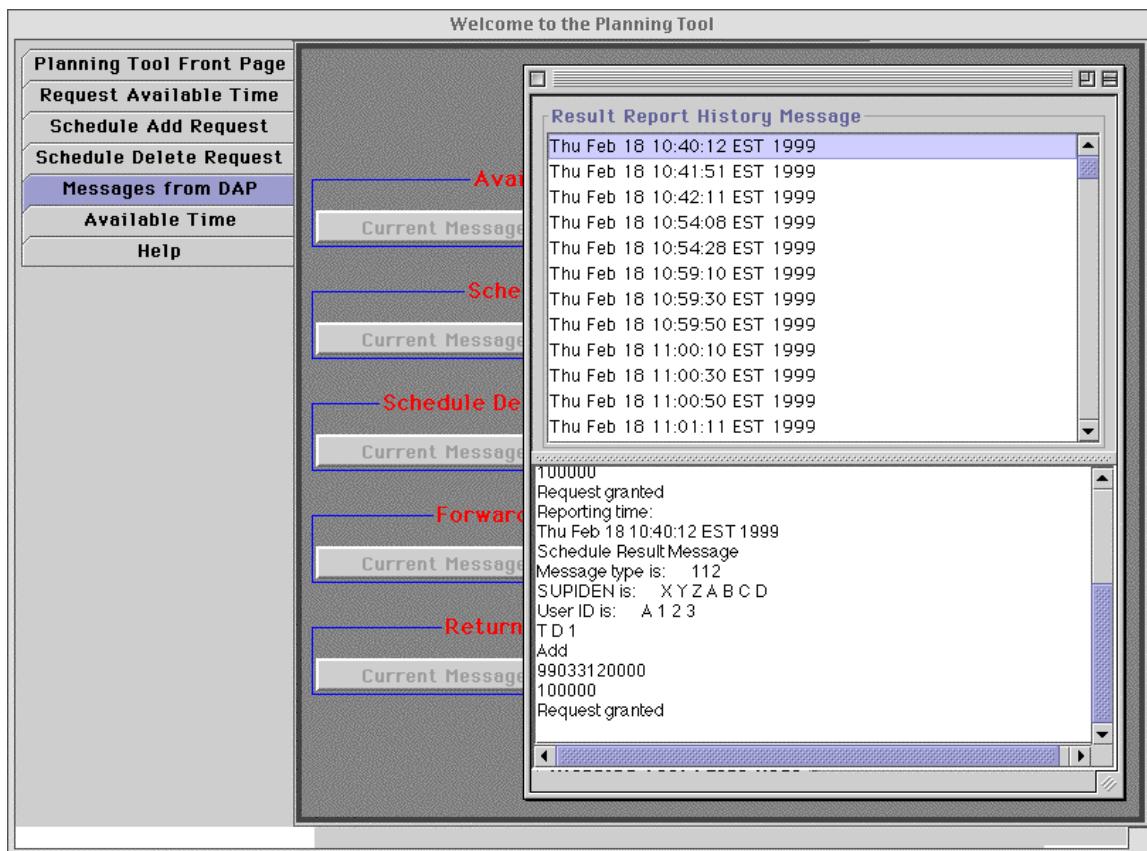


Figure 3-7: Schedule Result Message (History Message) Report Form

### 3.1.6 Available Time Report Form

The Available Time Report Form shown in Figure 3-8 is selected when a PT user needs to check current available time or events. This form displays the calendar and the scheduling table. PT users can check scheduling information for two months (Current month and the following month) for up to 5 TDRS satellites. The table at the bottom part of the screen displays the 24-hour scheduling events for the selected week and TDRS satellite number. For instance, this screen shows the scheduling events for TDRS1 in the first week of February 1999. If a PT user wants to check a scheduling event and time slot for a specific date, the user can select a table column and a detailed pie chart for the times will be displayed as shown in Figure 3-9. A pie chart displays the details for the selected date and TDRS satellite. White parts of the pie chart show the occupied time slots, while the black parts show the available time slots for the specific TDRS.

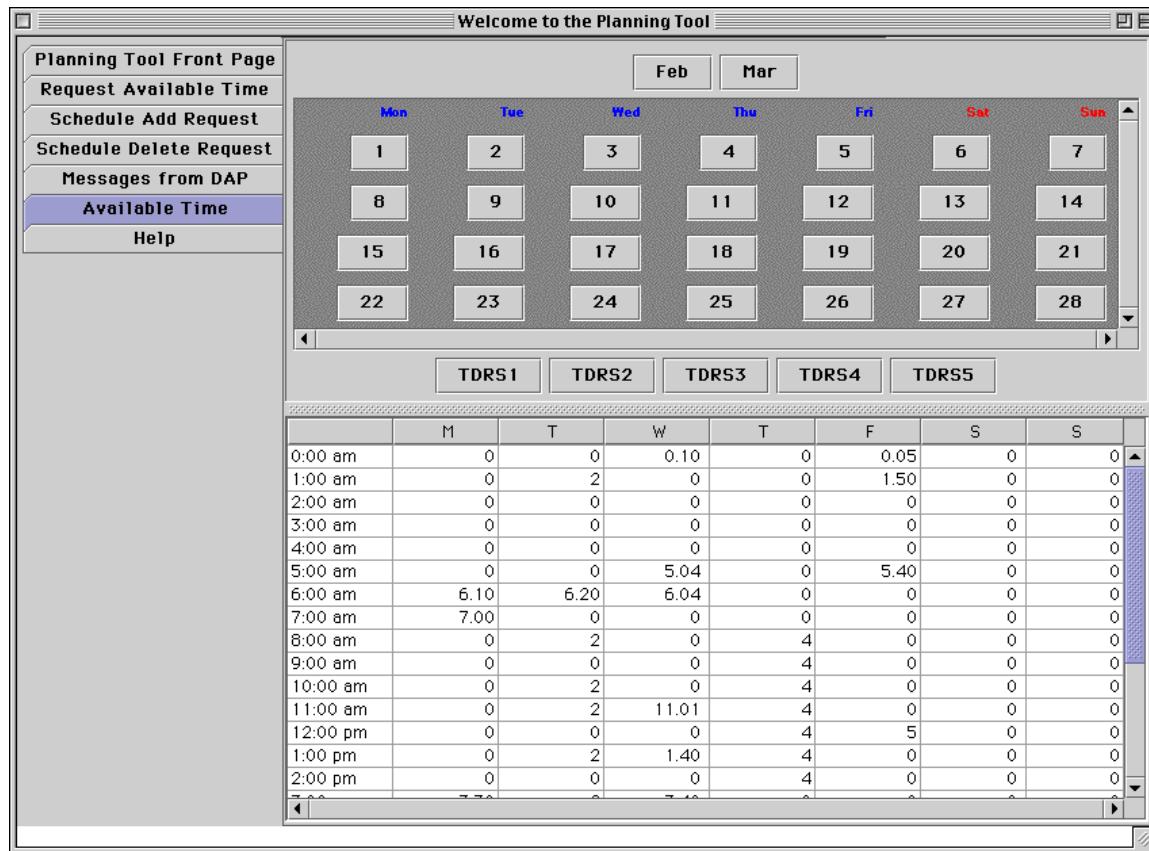
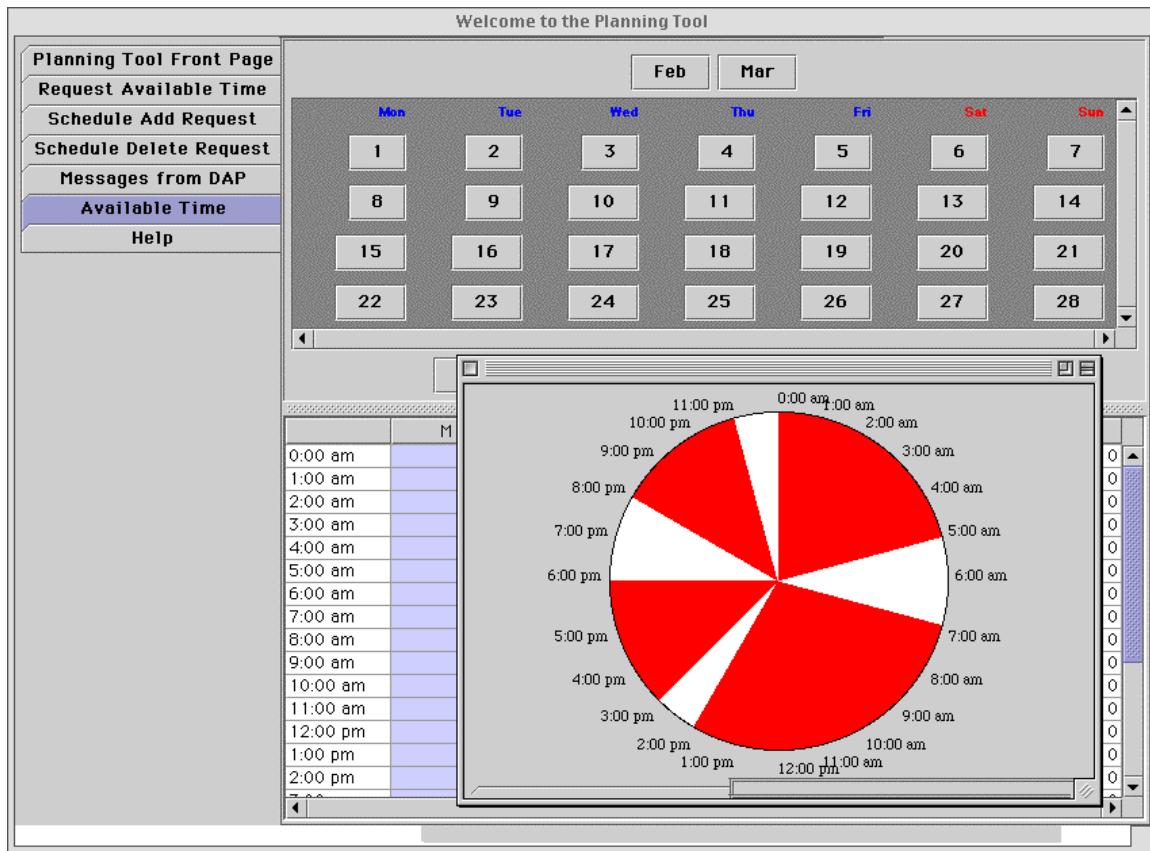


Figure 3-8: Available Time Report Form



**Figure 3-9: Available Time Pie Chart Report Form**

### 3.2 Demand Access Planning Tool PDL

The following shows the high level description of the Phase 1 Planning Tool program, seven classes and their relations are defined in Figure 3-10.

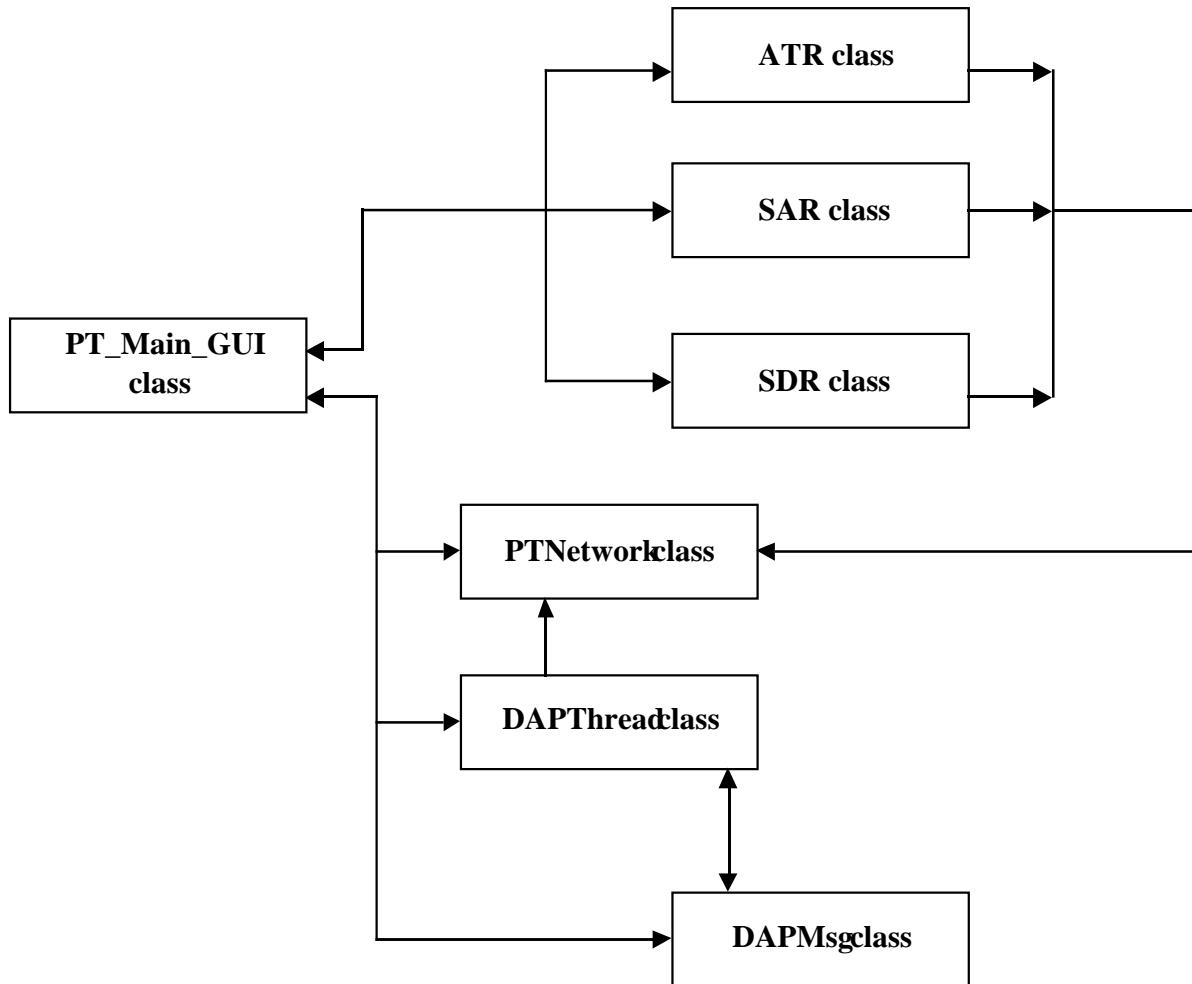


Figure 3-10: Planning Tool Classes

### 3.2.1 PT\_Main\_GUI

```

public class PT_Main_GUI
begin

define GUI Menu Items;
define Control Buttons;
define DAP Messages length in bytes;
define DAP message buffers ;

define PT_Main_GUI constructor;
define menus or buttons or other GUI components;
handle User Interface choice event;

process GUI Components choices
begin
    if GUI-ATR-msg then call ATR class ;

    else if GUI-SAR-msg then call SAR class;
    else if GUI-SDR-msg then call SDRprocess;
    .....
    else if GUI-AT-msg then call DAPMsg class to process AT;
    else if GUI-SR-msg then call DAPMsg class to process SR;
    else if GUI-FSD-msg then call DAPMsg class to process FSD;
    else if GUI-RSD-msg then call DAPMsg class to process RSD;
    else if GUI-SDN-msg then call DAPMsg class to process SDN;
    return true when a GUI component action is processed
    return false otherwise
end

handle close main PT_GUI window event;

main program, initiates the whole program

begin
    readDAP IP address and port numbers from a configuration file;
    start PTNetwork thread;
    Start Read-DAP-message thread;

    end
end

```

### 3.2.2 DAPThread

```

public class DAPThread
begin

while true do
begin

    Connect to DAP IP address and Port number;
    if DAP is down, retry the network connection until the connection is up;
    if a DAP message comes in then record the report coming time ;

    // store DAP messages into buffer in binary form;

    if msgtype is AT & msglength is ok then put AT msg in buffer; // Available Time
message
        if msgtype is SRM & msglength is ok then put SRM msg in buffer; // Schedule Result
message
        if msgtype is FSD & msglength is ok then put FSD msg in buffer; // Forward Service
message
        if msgtype is RSD & msglength is ok then put RSD msg in buffer; // Return Service
message
        if msgtype is SDN & msglength is ok then put SDN msg in buffer; // Schedule Deletion
Notification message

        save message into history file if the message length is correct;
end while
end

```

### 3.2.3 PTNetwork

```

public class PTNetwork
begin
    keep getting DataInputStream on the socket;
    keep getting DataOutputStream on the socket;
    if socket==null then keep retrying to connect;

    upon call, send byte data (a PT message) to the outputstream;
    if InputDataStream !=null then get byte data (a DAP message) from the inputstream;
end;

```

### 3.2.4 ATR-process

```
public class ATR-process
begin
    interface layout for the ATR message
    define ATR GUI layout;
    define ATR layoutmanager;
    handle button and menu choices;
    handle close ATR GUI window event;
    convert ATR message into a byte array,
    call PT-network class to send the message to DAP;
end;
```

### 3.2.5 SAT-process

```
public class SAT-process
begin
    interface layout for the SAR message
    define SAR GUI layout;
    define SAR layoutmanager;
    handle button and menu choices;
    handle close SAR GUI window event;
    convert SAR message into a byte array,
    call PT-network class to send the message to DAP;
end;
```

### 3.2.6 SDR-process

```
public class SDR-process
begin
    interface layout for the SDR message
    define SDR GUI layout;
    define SDR layoutmanager;
    handle button and menu choices;
    handle close SDR GUI window event;
    convert SDR message into a byte array,
    call PT-network class to send the message to DAP;
end;
```

### 3.2.7 DAPMsg\_process

```
public class DAPMsg_process
begin

    define DAP messages interface components;
    define DAP messages interface layout manager;
    handle close window event;
    handle button and menu choices
    begin
        if AT-GUI choice then
            begin
                parse AT message saved in the buffer;
                display AT message;
                save AT message;
            end

        if SR-GUI choice then
            begin
                parse SR message saved in the buffer;
                display SR message;
                save SR message;
            end

        if FSD-GUI choice then
            begin
                parse FSD message saved in the buffer;
                display FSD message;
                save FSD message;
            end

        if RSD-GUI choice then
            begin
                parse RSD message saved in the buffer;
                display RSD message;
                save RSD message;
            end

        if SDN-GUI choice then
            begin
                parse SDN message saved in the buffer;
                display SDN message;
                save SDN message;
            end
    end
end
```

### 3.3 Message Formats for the DAP to Planning Tool Interface

Planning Tool (PT) communicates with DAP using TCP/IP protocol. PT clients establish and maintain connection to DAP on an as-needed basis. PT clients may terminate their services at any time and the DAP will not be interrupted by PT's termination. PT clients are responsible for re-establishing connections to DAP if DAP process terminates abnormally.

Messages between DAP and PT are shown in Figure 3-11. PT can send three types of messages to DAP and DAP can send five types of messages to PT. Messages sent from PT and their corresponding message(s) received from DAP are shown as paired arrows in Figure 3-11. For instance, PT sends a Schedule Add Request Message (requesting Forward service) to DAP, it then will receive a Schedule Result Message which indicates the time scheduled for the service as well as the Forward Service Data Message that shows detailed information for the forward service.

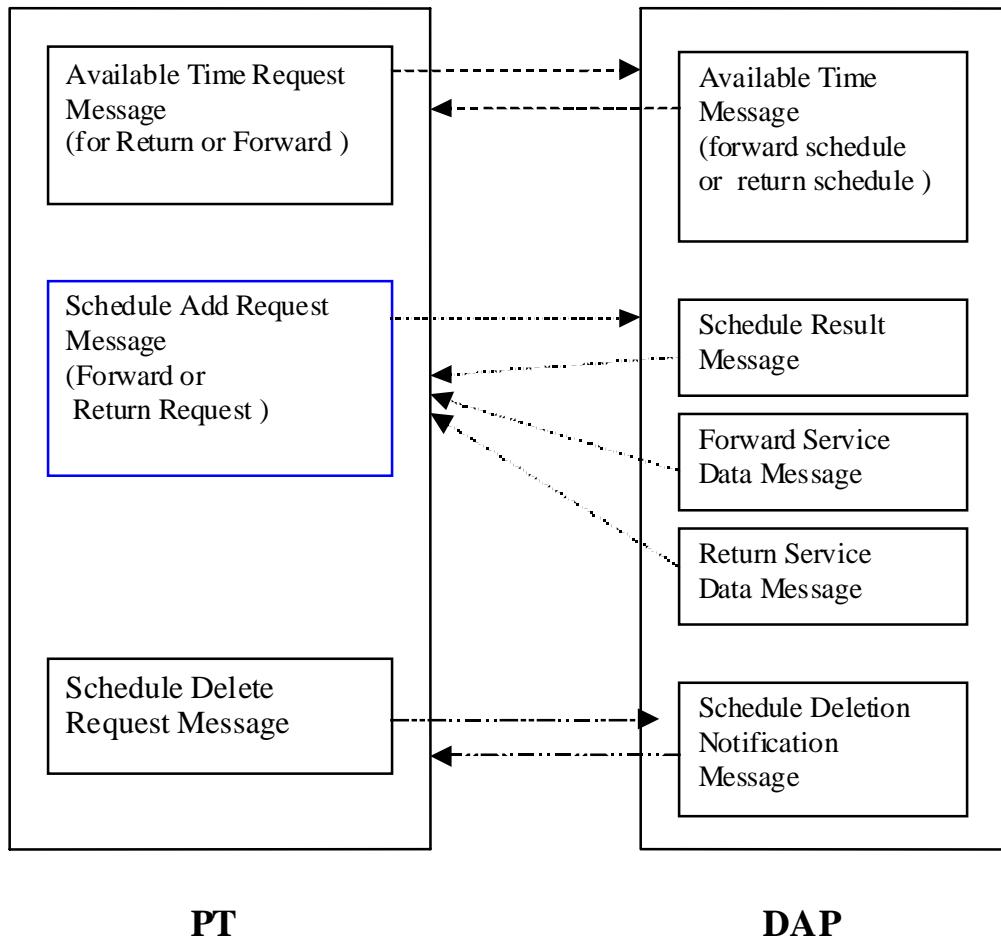


Figure 3-11: Messages between PT and DAP

### 3.3.1 Available Time Request Message (From PT to DAP) (Msg 101)

The ATR message allows the PT to request the DAP possible available time for a TDRS service. The format of this message is shown in Table 3-1.

**Table 3-1. Available Time Request Message Format**

Item #	# of Bytes	Data Type	Bit Num.	Data Field	Description
1	2	short Integer	0-15	Message Type	101—Available Time Request Msg 102—Schedule Add Request Msg 103—Schedule Delete Request Msg
2	8	Char	0-63	User ID	Requester ID (four ASCII characters)
3	1	Enum. Type	0	Normal/Emergency Support Flags	0 = Normal support 1 = Emergency support
			1-7	Spares	
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	6	String	0-47	Service Date	DDDMMMYY (day, month, and year, request available time on a specific date)
6	1	Enum. Type	0	Service Type	0=forward 1=return
7		Enum. Type	1-3	DAR User	00=not applicable in case of forward service request 01=shared 02=dedicated
8		Enum. Type	4-7	equip ID	00=not applicable in case of forward service request IBUG ID (01-05)

### 3.3.2 Schedule Add Request Message ( From PT to DAP) (Msg 102)

The SAR message allows the PT to request the DAP to add a forward or return service schedule event for a TDRS. The message format is shown in Table 3-2. The following rule also applies to the SAR message:

- All related services (forward, return, etc) for one SUPIDEN for one TDRS for one contiguous time interval will be contained in the same SAR. However, if needed to obtain the necessary support the PT may submit two or more non-conflicting SARs applicable to the same TDRS at the same time.
- If need to modify the schedule request, a Schedule Delete Request message needs to be sent, then a new SAR is sent to contain the new (modified) request.

**Table 3-2. Schedule Add Request Message Format**

Item #	# of Bytes	Data Type	Start Byte	Data Field	Description
1	2	short Integer	0-15	Message Type	101—Available Time Request Msg 102—Schedule Add Request Msg 103—Schedule Delete Request Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	8	Char	0-63	User ID	Requester ID (four ASCII characters)
4	1	Enum. Type	0	Normal/Emergency Support Flags	0 = Normal support 1 = Emergency support
			1-7	Spares	
5	6	Char	0-23	TDRS	AAA (three alphanumeric characters)
6	1	Enum. Type	0	Service Type	0=forward 1=return
7		Enum. Type	1-3	DAR User	00—not applicable in case of forward service request 01=shared 02=dedicated
8		Enum. Type	4-7	equip ID	00—not applicable in case of forward service request IBUG ID (01-05)
9	11	String	0-87	Service Start Time	YYDDDHHMMSS (year, number of days in the year, hour, minute, seconds)
10	6	String	0-47	Service Duration	HHMMSS (hour, minute, seconds)
11	4	Integer	0-31	Number of Parameters to be changed	refers to parameters of the configuration code to be changed

### 3.3.3 Schedule Delete Request Message (From PT to DAP) (Msg 103)

The SDR message allows PT to request the DAP to delete a scheduled. The message format is shown in Table 3-3. The following rule applies to the SDR message:

- Changes to configurations require the entire scheduled event to be deleted via a SDR, followed by a SAR submitted to schedule an event.

**Table 3-3. Schedule Delete Request Message Format**

Item #	# of Bytes	Data Type	Start Byte	Data Field	Description
1	2	short Integer	0-15	Message Type	101—Available Time Request Msg 102—Schedule Add Request Msg 103—Schedule Delete Request Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	8	Char	0-63	User ID	Requester ID (four ASCII characters)
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	1	Enum. Type	0	Service Type	0=forward 1=return
		Bit	1-7	Spares	
6	11	String	0-87	Old Event Start Time	YYDDDHMMSS
7	6	String	0-47	Old Event Duration	HHMMSS (hour, minute, seconds)

### 3.3.4 Available Time Message ( From DAP to PT) (Msg 111)

The AT message sent from DAP to PT lists all the available time for Return or Forward Services as requested in the Available Time Request Message within minus 3 and plus 3 days of Service Date specified in ATR message. Table 3-4 gives the AT message format.

**Table 3-4. Available Time Message Format**

Item #	# of Bytes	Data Type	Start Byte	Data Field	Description
1	2	short Integer	0-15	Message Type	111—Available Time Msg 112—Schedule Result Msg 113—Schedule Deletion Notification Msg 114—Forward Service Data Msg 115—Return Service Data Msg
2	8	Char	0-63	User ID	Requester ID (four ASCII characters)
3	1	Enum. Type	0	Normal/Emergency Support Flags	0 = Normal support 1 = Emergency support
		Bit	1-7	Spares	
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	1	Enum. Type	0	Service Type	0=forward 1=return
6		Enum. Type	1-3	DAR User	00=not applicable in case of forward service request 01=shared 02=dedicated
7		Bit	4-7	equip ID	00=not applicable in case of forward service IBUG ID (01-05)
8	2	Short Integer	0-8	Number of total time slots available	
9	11	String	0-687	Service Start Time-1	YYDDDHHMMSS
10	11	String	0-87	Service Stop Time-1	YYDDDHHMMSS
				..... (other slots, as in field 9 and field 10)	

### 3.3.5 Schedule Result Message ( From DAP to PT) (Msg112)

The SR message is sent from DAP to PT when a schedule is issued in response to a SAR. The format is shown in Table 3-5.

**Table 3-5. Schedule Result Message Format**

Item #	# of Bytes	Data Type	Start Byte	Data Field	Description
1	2	short Integer	0-15	Message Type	111—Available Time Msg 112—Schedule Result Msg 113—Schedule Deletion Notification Msg 114—Forward Service Data Msg 115—Return Service Data Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	8	Char	0-63	User ID	Requester ID (four ASCII characters)
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	1	Enum. Type	0-1	Reference Action	10= Add, 11=Delete
		Bit	2-7	Spares	
6	11	String	0-87	Service Start Time	YYDDDHMMSS
7	6	String	0-47	Service Duration	HHMMSS (hour, minute, seconds)
8	4	Integer	0-31	Result Code	refer to result code table ( <b>Table 3-6</b> )

**Table 3-6. Result Code Table**

Code	Result
00	Request granted(only used in the active time frame)
01	Request canceled by the operator
02	Request rejected due to conflict
03	Request received and spooled for processing
04	Request placed in valid request queue

### 3.3.6 Schedule Deletion Notification Message ( From DAP to PT) (Msg 113)

When a scheduled event is deleted in response to a deletion request message, DAP sends a SDN message to PT. The format of this message is shown in Table 3-7.

**Table 3-7. Schedule Deletion Notification Message Format**

Item #	# of Bytes	Data Type	Start Byte	Item Name	Description
1	2	short Integer	0-15	Message Type	111—Available Time Msg 112—Schedule Result Msg 113—Schedule Deletion Notification Msg 114—Forward Service Data Msg 115—Return Service Data Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	8	Char	0-63	User ID	Requester ID (four ASCII characters)
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	1	Enum. Type	0	Service Type	0=forward 1=return
6		Enum. Type	1-3	DAR User	00=not applicable in case of forward service request 01=shared 02=dedicated
7		Bit	4-7	equip ID	00=not applicable in case of forward service IBUG ID (01-05)
8	11	String	0-87	Service Start Time	YYDDDHMMSS
9	6	String	0-47	Service Duration	HHMMSS (hour, minute, seconds)
10	1	Bit	0	Deletion Status	00 = incomplete, 01 = complete
		Bit	1-7	Spares	

### 3.3.7 Forward Service Data Message (From DAP to PT) (Msg 114)

The FSD message consists of PT customer schedule information and the fixed and configurable parameters for MA Forward service. This message is shown in Table 3-8.

**Table 3-8. Forward Service Data Message Format**

Item #	# of Bytes	Data Type	Start Byte	Name	Description
1	2	short Integer	0-15	Message Type	111—Available Time Msg 112—Schedule Result Msg 113—Schedule Deletion Notification Msg 114—Forward Service Data Msg 115—Return Service Data Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	6	Char	0-63	User ID	Requester ID (four ASCII characters)
4	6	Char	0-63	TDRS	AAA (three alphanumeric characters)
5	11	String	0-87	Service Start Time	YYDDDHMMSS
6	11	String	0-87	Service Stop Time	YYDDDHMMSS

### 3.3.8 Return Service Data Message Format ( From DAP to PT) (Msg 115)

The RSD message consists of PT customer schedule information and the fixed and configurable parameters for MA Return service. This message is shown in Table 3-9.

**Table 3-9. Return Service Data Message Format**

Item #	# of Bytes	Data Type	Start Byte	Name	Description
1	2	short Integer	0-15	Message Type	111—Available Time Msg 112—Schedule Result Msg 113—Schedule Deletion Notification Msg 114—Forward Service Data Msg 115—Return Service Data Msg
2	14	Char	0-111	SUPIDEN	Refer to 534-808
3	8	Char	0-63	User ID	Requester ID (four ASCII characters)
4	6	Char	0-47	TDRS	AAA (three alphanumeric characters)
5	1	Bit	0-2	DAR User	00=not applicable in case of forward service request 01=shared 02=dedicated
6		Bit	3-6	equip ID	00=not applicable in case of forward service request IBUG ID (01-05)
		Bit	7	Spares	
7	11	String	0-87	Service Start Time	YYDDDHMMSS
8	11	String	0-87	Service Stop Time	YYDDDHMMSS

**Note:**

- In Java, the length of some of the data types may be different from what is defined in other languages.
 

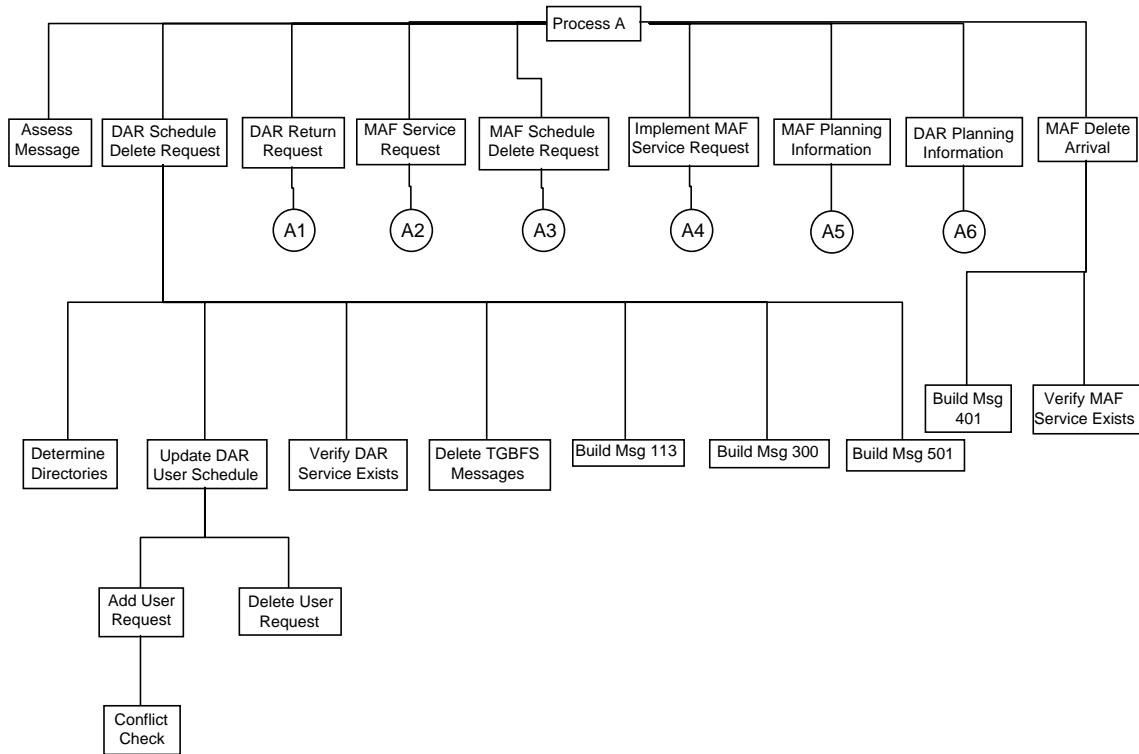
Integer	4 bytes
Char	2 bytes
Short	2 bytes
Long Integer	8 bytes
Double	8 bytes
- Message numbers 100-199 are reserved for messages sent between PT and DAP. Message numbers 101-103 for messages sent from PT to DAP, and starting at 111 through 115 for messages from DAP to PT. Messages 104 to 109 are reserved for future PT to DAP messages.

## 4. DAP Process A Service Manager Detailed Design

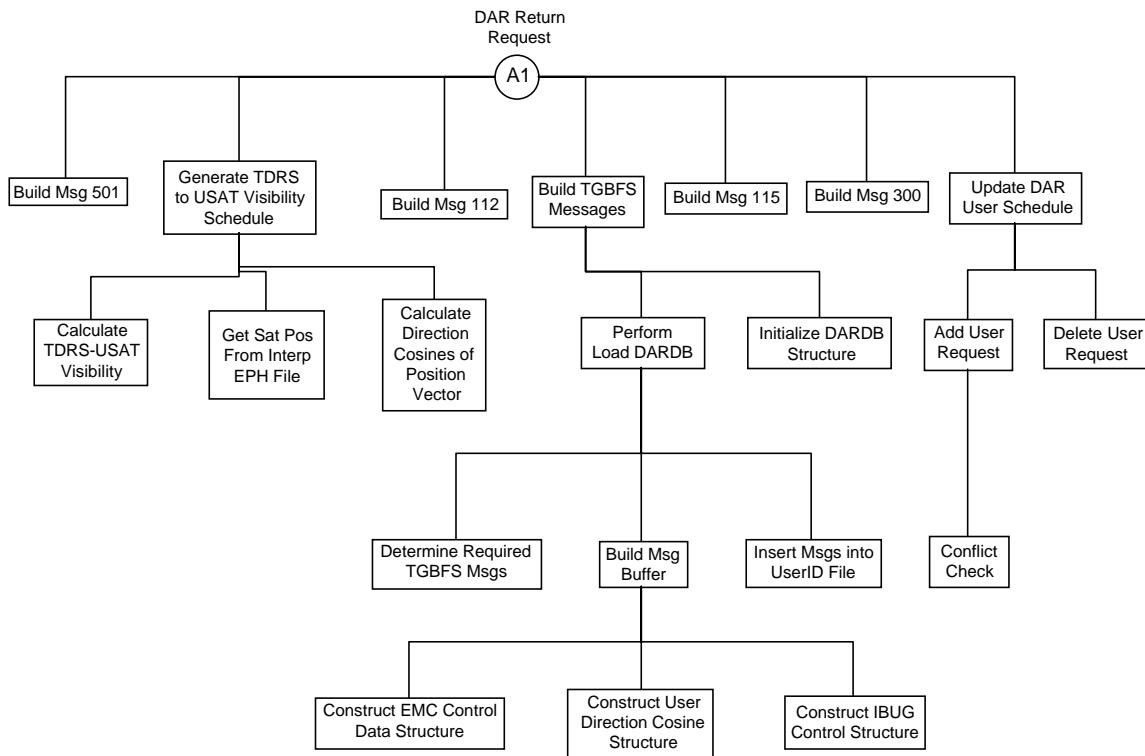
The following assumptions are made for Process A functions during Phase 1 implementation:

- Process A maintains the DAR DB (See Figures 6-2, 6-3, and Tables 6-1 and 6-2 from Process C documentation) and Process C reads it
- Process A notifies Process C when a change has been made to the DAR DB
- The shared beamformer procedure will be implemented in Phase 2
- Process A handles requests both for MAF and for DAR service requests and deletions
- Process A does not change the working TUT when new MAF service requests are implemented, but alerts Process B to changes which need to be made
- When MAF or DAR services are scheduled or deleted, Process A sends relevant commands to DAB Emulator
- Process A determines resource availability for MAF, but uses the NCC emulator to implement scheduling
- The NCC emulator is not used for DAR scheduling. Process A notifies Process C of return services after Process A has modified the DAR database
- A user's visibility schedule determines when the user can see the TDRSS satellites
- A user's availability schedule determines when service is available for a user using the TDRSS satellites.

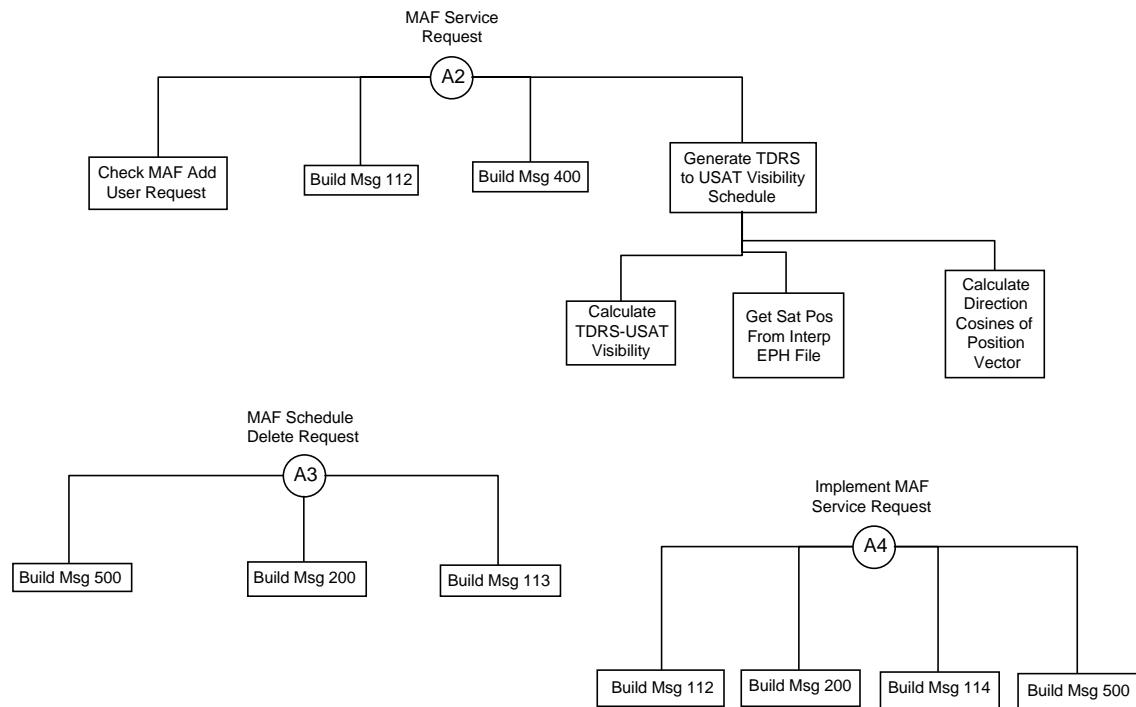
Figure 4-1 shows the structure chart for functions of Process A during Phase 1 implementation.



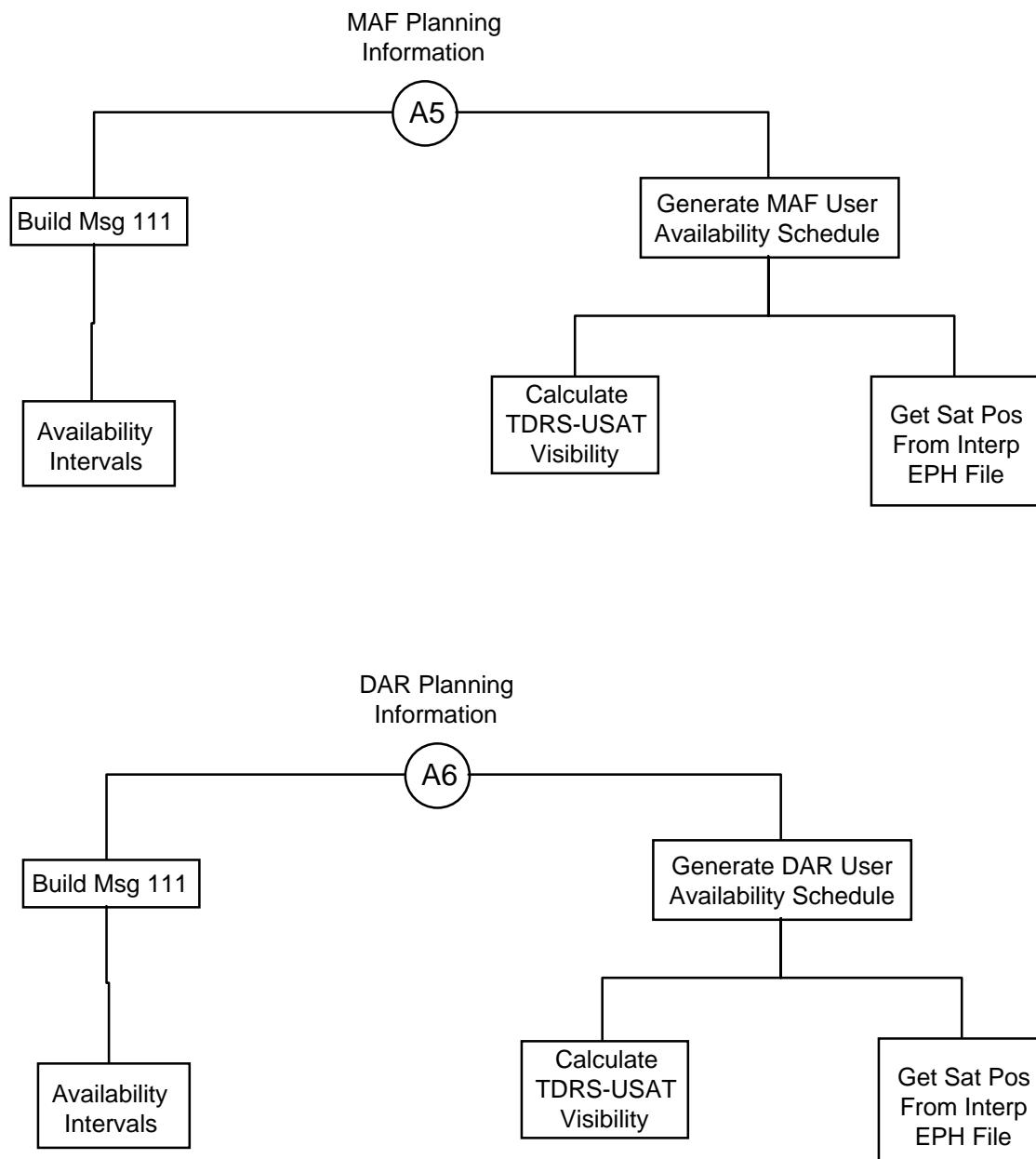
**Figure 4-1: DAP Phase 1 Procedure hierarchies for Process A**



**Figure 4-1: DAP Phase 1 Procedure hierarchies for Process A (continued)**



**Figure 4-1: DAP Phase 1 Procedure hierarchies for Process A (continued)**



**Figure 4-1: DAP Phase 1 Procedure hierarchies for Process A (continued)**

## 4.1 PDL Procedures for Process A

The following subsections describe the PDL for Process A functions during Phase 1 implementation.

### 4.1.1 Begin Process A

```

Begin Process A
    Initialize process
    // Endless main processing loop
    Do while TRUE
        // Wait for new messages from PT
        Wait for message
        If message arrives
            Get Msg ID
            Perform Procedure Assess Message
                // Take the appropriate action
                Case (ID)
                    103A: Procedure MAF Delete Arrival
                    103B: Procedure DAR Schedule Delete Request
                    101A: Procedure MAF planning information
                    101B: Procedure DAR planning information
                    102A: Procedure MAF service request
                    102B: Procedure DAR return request
                    403: Procedure MAF Schedule Delete Request
                    402: Procedure Implement MAF Service Request
                    2000: Print out error message
                End Case
            End if // message arrives
        End do while
    End Process A

```

### 4.1.2 Procedure MAF Delete Arrival

```

Procedure MAF Delete Arrival
    Procedure Verify MAF Service Exists
    If NoService then
        Procedure Build Msg 113
        Send Msg 113 to PT // delete MAF error message
    Else
        Procedure Build Msg 401
        Send Msg 401 to NCC
    End else
End Procedure MAF Delete Arrival

```

### 4.1.3 Procedure Assess Message

Procedure Assess Message

```
ID = 2000 // Error Code
If Msg ID = 103 and Service Type = Forward
    ID = 103A
End if
Else if Msg ID = 103 and Service Type = Return
    ID = 103B
End if
Else if Msg ID = 101 and Service Type = Forward
    ID = 101A
End if
Else if Msg ID = 101 and Service Type = Return
    ID = 101B
End if
Else if Msg ID = 102 and Service Type = Forward
    ID = 102A
End if
Else if Msg ID = 102 and Service Type = Return
    ID = 102B
End if
Else if Msg ID = 403
    ID = 403
End if
Else if Msg ID = 402
    ID = 402
End if
End Procedure Assess Message
```

#### 4.1.4 Procedure MAF Schedule Delete Request

Procedure MAF Schedule Delete Request

    Procedure Build Msg 200

    Send Msg 200 to Process B

    Procedure Build Msg 113

    Send Msg 113 to PT

    Procedure Build Msg 500

    Send Msg 500 to DAB

End Procedure MAF Schedule Delete Request

#### 4.1.5 Procedure Verify MAF Service Exists

Procedure Verify MAF Service Exists

    NoService = FALSE

    Open MAF TUT File

    Do while (not EOF for MAF TUT File) and (NoService = FALSE)

        Read record and get t<sub>START\_TUT</sub> and t<sub>STOP\_TUT</sub>

        If (t<sub>START\_TUT</sub> >= t<sub>START\_DELETE</sub>) and (t<sub>START\_TUT</sub> <= t<sub>STOP\_DELETE</sub>)

            or (t<sub>STOP\_TUT</sub> >= t<sub>START\_DELETE</sub>) and (t<sub>STOP\_TUT</sub> <= t<sub>STOP\_DELETE</sub>)

            NoService = TRUE

    End if

    End do while

    Close MAF TUT File

End Procedure Verify MAF Service Exists

#### **4.1.6 Procedure DAR Schedule Delete Request**

```

Procedure DAR Schedule Delete Request
  Perform Verify DAR Service Exists
  If NoService then
    Procedure Build Msg 113
    Send Msg 113 to PT // delete DAR error message
  Else
    Get UserID from delete service request
    Perform Determine Directories
    Perform Delete TGBFS Messages
    OperationType = Delete
    Perform Update DAR User Schedule
    Procedure Build Msg 300
    Send Msg 300 to Process C
    Procedure Build Msg 113
    Send Msg 113 to PT
    Procedure Build Msg 501
    Send Msg 501 to DAB
  End if
End Procedure DAR Schedule Delete Request

```

#### **4.1.7 Procedure Verify DAR Service Exists**

```

Procedure Verify DAR Service Exists
  NoService = TRUE
  Open User Schedule File
  Do while (not EOF for User Schedule File) and (NoService = TRUE)
    Read record and get t START_SERV and t STOP_SERV
    If (t START_SERV = t START_DELETE) and (t STOP_SERV = t STOP_DELETE) then
      NoService = FALSE
    End if
  End do while
  Close User Schedule File
End Procedure Verify DAR Service Exists

```

#### 4.1.8 Procedure Determine Directories

```

Procedure Determine Directories
  Open Temporary Directory File
  t = t START_DELETE
  Do while (t <= t STOP_DELETE )
    Convert t to ASCII string T
    Truncate minute and seconds bytes from T // implementation note: last 4 bytes
    Write T to the next Temporary Directory File Record // these are the directory names
    t = t + 1 hour
  End do while
  Close Temporary Directory File
End Procedure Determine Directories

```

#### 4.1.9 Procedure Delete TGBFS Messages

```

Procedure Delete TGBFS Messages
  Open Temporary Directory File // Temporary Directory File ≡ TDF
  Do while (not EOF for TDF) // move to next directory in the delete request window
    Read record TDFREC from TDF
    Get T from TDFREC
    Open UserID File // UserID File ≡ UF and is located in /DABDR/T/UserID
    Open Temporary UserID File // Temporary UserID File ≡ TUF
    Read record UFREC from UF
    Get t_UFREC from UFREC
    // save the message data outside the delete request window
    Do while (not EOF for UF) // step through the entire UF
      If [(t_UFREC < t START_DELETE ) or (t_UFREC > t STOP_DELETE )] then
        Write UFREC to TUF
      End if
      If (not EOF for UF)
        Read record UFREC from UF
        Get t_UFREC from UFREC
      End if
    End do while
    Close Temporary User File
    Close UserID File
    Copy TUF to UF // keep left over message data for other DAR requests
  End do while
  Close Temporary Directory File
End Procedure Delete TGBFS Messages

```

#### 4.1.10 Procedure MAF service request

Procedure MAF service request

```
    Procedure Generate TDRS-USAT Visibility Schedule
    If ScheduleOK = TRUE
        Extract TUT Schedule from the TUT DB
        Procedure Check MAF Add User Request
        If NoConflict = TRUE
            Procedure Build Msg 400
            Send Msg 400 to NCC
        Else
            Procedure Build Msg 112
            Send Msg 112 to PT
        End if
    Else
        Procedure Build Msg 112
        Send Msg 112 to PT
    End if
End Procedure MAF service request
```

#### 4.1.11 Procedure Implement MAF Service Request

Procedure Implement MAF Service Request

```
    Procedure Build Msg 200
    Send Msg 200 to Process B
    Procedure Build Msg 112
    Send Msg 112 to PT
    Procedure Build Msg 114
    Send Msg 114 to PT
    Procedure Build Msg 500
    Send Msg 500 to DAB
End Procedure Implement MAF Service Request
```

#### **4.1.12 Procedure DAR return request**

```

Procedure DAR return request
    Procedure Generate TDRS-USAT Visibility Schedule
        If ScheduleOK = TRUE
            Procedure Update DAR User Schedule
            If (Conflict = FALSE) and (AddError = FALSE)
                Procedure Build TGBFS Messages
                Procedure Build Msg 300
                Send Msg 300 to Process C
                Procedure Build Msg 112
                Send Msg 112 to PT
                Procedure Build Msg 501
                Send Msg 501 to DAB
            Else
                Procedure Build Msg 112
                Send Msg 112 to PT
            End if
        Else
            Procedure Build Msg 112
            Send Msg 112 to PT
        End if
    End Procedure DAR return request

```

#### **4.1.13 Procedure Build TGBFS Messages**

```

Procedure Build TGBFS Messages
    Open Temporary Direction Cosine File // contains the raw data for TGBFS Msg Type 3
    Obtain t FIRST from first record in Temporary Direction Cosine File
    Obtain t LAST from first record in Temporary Direction Cosine File
    Get UserID
    Perform Initialize DARDB Structure
    Perform Load DARDB
    Close Temporary Direction Cosine File
    Delete Temporary Direction Cosine File
End Procedure Build TGBFS Messages

```

#### 4.1.14 Procedure Load DARDB

```
Procedure Load DARDB
  Open Temporary Directory File
  FirstMsg = TRUE
  Do while (not EOF for Temporary Directory File)
    Read record from Temporary Directory File
    Convert ASCII string T to numeric tD // T is the ASCII directory name
    Continue = TRUE
    Do while (Continue) and (not EOF for Temporary Direction Cosine File)
      Read next record from the Temporary Direction Cosine File
      Get the time tDC from the record
      Open Temporary Message File // open as empty file
      If (tD <= tDC) and (tD + 1 hour < tDC) then
        Perform Determine Required TGBFS Msgs
        Perform Build Msg Buffer
        Write buffer MsgBuf to Temporary Message File
        FirstMsg = FALSE
      Else
        Continue = FALSE
        Move Temporary Direction Cosine file pointer back one record
      End if
      Open UserID File in directory T // implementation note: /DARDB/T/UserID
      Perform Insert Msgs into UserID File
      Close UserID File in directory T
      Close Temporary Message File // empty file
    End do while
  End do while
  Close Temporary Directory File
End Procedure Load DARDB
```

#### 4.1.15 Procedure Initialize DARDB Structure

```

Procedure Initialize DARDB Structure
  Open Temporary Directory File
  t = t FIRST
  Do while (t <= t LAST )
    Convert t to ASCII string T // implementation note: last 4 bytes
    Truncate minute and seconds bytes from T
    Compare T to DARDB subdirectory names // implementation note: system ('ls /DARDB/T');
    If T does not exist then // if T exists the UserID File within exists also
      Compare UserID // implementation note: system ('ls /DARDB/T/UserID');
      If UserID File does not exist then
        Create empty UserID File
      End if
    End if
    Write T to the next Temporary Directory File Record
    t = t + 1 hour
  End do while
  Close Temporary Directory File
End Procedure Initialize DARDB Structure

```

#### 4.1.16 Procedure Determine Required TGBFS Msgs

```

Procedure Determine Required TGBFS Msgs
// logic to decide what messages to send at t DC
  MsgMap = 0 // zero out all bits - bit number corresponds to TGBFS message number
  If FirstMsg = TRUE then // initialization messages for the beamformer
    Bit 0 of MsgMap = 1 // EMC Control Data Message
    Bit 3 of MsgMap = 1 // User Direction Cosine Message
    Bit 4 of MsgMap = 1 // IBUG Control Data Message
    Bit 31 of MsgMap = 1 // First message flag
  Else // non initialization
    Bit 3 of MsgMap = 1 // User Direction Cosine Message
  End if
End Procedure Determine Required TGBFS Msgs

```

#### 4.1.17 Process Build Msg Buffer

Process Build Msg Buffer

Null 808 byte MsgBuf structure

Put  $t_{DC}$  into field 1 of MsgBuf // **message time for group segment (see Process C)**

Put MsgMap into field 2 of MsgBuf

If FirstMsg = TRUE then

    Perform Construct EMC Control Data Structure

    Put EMC Control Data Structure into field 3 of MsgBuf // see **Process C for file record**

**layout**

    Perform Construct User Direction Cosine Structure

    Put EMC User Direction Cosine into field of 6 MsgBuf

    Perform Construct IBUG Control Structure

    Put IBUG Control Data Structure into field of 7 MsgBuf

Else

    Perform Construct User Direction Cosine Structure

    Put EMC User Direction Cosine into field of 6 MsgBuf

End if

End Process Build Msg Buffer

#### 4.1.18 Procedure Insert Msgs into UserID File

Procedure Insert Msgs into UserID File

Get size of UserID File // **note: /DARDB/T/UserID and UserID File ≡ UF**

If UserID File is empty then

    Copy Temporary Message File to UserID File // **Temporary Message File ≡ TMF**

Else // **Insert Temporary Message File records into the appropriate place in the UserID File**

    Open Temporary UserID File // **Temporary UserID File ≡ TUF**

    Read the first record TMFREC in the TMF

    Read the first record UFFREC in the UF

    If (t TMFREC < t UFFREC ) // **insert the TMF in front of all the existing UF records in TUF**

        Write TMFREC to TUF

        Do while (not EOF for TMF)

            Read TMFREC from TMF

            Write TMFREC to TUF

            End while

            Write UFREC to TUF

            Do while (not EOF for UF)

                Read UFREC from TMF

                Write UFREC to TUF

            End while

            Close TUF

            Copy TUF to UF // **overwrite existing UF with TUF**

            Delete TUF

    Else // **insert TMF records at intermediate point in UF records or after UF records**

        Read UFREC from UF

        Read TMFREC from TMF

        Write UMFREC to TUF

        Do while (t TMFREC > t UMFREC ) // **insert UF records in TUF until TMF record is to be inserted**

            Read UFREC from UF

            Write UFREC to TUF

            End do while

            Write TMFREC to TUF

            Do while (not EOF for TMF) // **insert TMF records in between or after the UF recs in TUF**

                Read TMFREC from TMF

                Write TMFREC to TUF

            End

            Do while (not EOF for UF) // **add remaining UF recs if TMF insertion was not at end of TUF**

                Read UFREC from UF

                Write UFREC to TUF

            End do while

            Close TUF

            Copy TUF to UF // **overwrite existing UF with TUF**

            Delete TUF

```
    End if
End if
End Procedure Insert Msgs into UserID File
```

#### **4.1.19 Procedure MAF Planning Information**

```
Procedure MAF Planning Information
    Procedure Generate MAF User Availability Schedule
    Build Msg 111
    Send Msg 111 to PT
End Procedure MAF Planning Information
```

#### **4.1.20 Procedure DAR Planning Information**

```
Procedure DAR Planning Information
    Procedure Generate DAR User Availability Schedule
    Build Msg 111
    Send Msg 111 to PT
End Procedure DAR Planning Information
```

#### 4.1.21 Procedure Generate TDRS-USAT Visibility Schedule

```

Procedure Generate TDRS-USAT Visibility Schedule
  Open USAT Interp EPH File // input file that exists
  Open TDRS Interp EPH File // input file that exists
  Open Temporary Direction Cosine File (Open New only) // output file
  AlwaysVisible = TRUE
  t BEGIN = t START_REQUEST // if odd sec, changed down to even sec
  t END = t STOP_REQUEST // if odd sec, changed up to even sec
  t CALC = t BEGIN
  Do while (AlwaysVisible = TRUE) and (t CALC <= t END)
    Get USAT position from USAT Interp EPH File
    Get TDRS position from TDRS Interp EPH File
    Procedure Calculate TDRS-USAT Visibility
    If (Visible) then // update Visibility Schedule
      If (DAR Service Request) then // update Direction Cosine File
        Procedure Calculate Direction Cosines of the Position Vector
        Write t CALC and Direction Cosines to Temporary Direction Cosine File
      End if
    Else
      AlwaysVisible = FALSE
    End if
    t CALC = t CALC + 2sec
  End do while
  Close Temporary Direction Cosine File
  Close USAT Interp EPH File
  Close TDRS Interp EPH File
// Indicate results of USAT-TDRS visibility check for decisions to follow
If AlwaysVisible = TRUE
  ScheduleOK = TRUE
Else
  ScheduleOK = FALSE
  Delete Temporary Direction Cosine File
End if
End Procedure Generate TDRS-USAT Visibility Schedule

```

#### 4.1.22 Procedure Generate MAF User Availability Schedule

```

Procedure Generate MAF User Availability Schedule
    Open USAT Interp EPH File // input file that exists
    Open TDRS Interp EPH File // input file that exists
    Open MAF TUT File // input file that exists
    Open MAF Visibility Schedule File // new output file, MVSF
    Open Temp TUT File // new output file
    Open MAF User Availability Schedule File // new output file, MUASF
    t BEGIN = t START_REQUEST // if odd sec, changed down to even sec
    t END = t STOP_REQUEST // if odd sec, changed up to even sec
    t CALC = t BEGIN
    Do while (t CALC <= t END)
        Get USAT position from USAT Interp EPH File
        Get TDRS position from TDRS Interp EPH File
        Procedure Calculate TDRS-USAT Visibility
        If (Visible) then
            Write "1" into MVSF
        Else
            Write "0" into MVSF
        End if
        WithinTUT = FALSE
        Go to beginning of MAF TUT file
        Do while (not EOF for MAF TUT file) and (WithinTUT = FALSE)
            Get next TUT
            If [(tCALC >= tSTART_TUT) and (tCALC <= tSTOP_TUT)]
                Set WithinTUT = TRUE
            End if
        End do while
        If (WithinTUT) then
            Write "1" into Temp TUT File
        Else
            Write "0" into Temp TUT File
        End if
        t CALC = t CALC + 2sec
    End Do while
    Close USAT Interp EPH File, TDRS Interp EPH File, MAF TUT File
    Go to beginning of MVSF and Temp TUT File
    Do while (not EOF of MVSF)
        Get next value from MVSF
        Get next value from Temp TUT File
        Multiply two values, write to MUASF
    End Do while
    Delete MVSF and Temp TUT File
    Close MUASF
End Procedure Generate MAF User Availability Schedule

```

#### 4.1.23 Procedure Generate DAR User Availability Schedule

```

Procedure Generate DAR User Availability Schedule
    Open USAT Interp EPH File // input file that exists
    Open TDRS Interp EPH File // input file that exists
    Open User Schedule File // input file that exists
    Open DAR Visibility Schedule File // new output file, DVSF
    Open Temp User Schedule File // new output file
    Open DAR User Availability Schedule File // new output file, DUASF
    t BEGIN = t START_REQUEST // if odd sec, changed down to even sec
    t END = t STOP_REQUEST // if odd sec, changed up to even sec
    t CALC = t BEGIN
    Do while (t CALC <= t END)
        Get USAT position from USAT Interp EPH File
        Get TDRS position from TDRS Interp EPH File
        Procedure Calculate TDRS-USAT Visibility
        If (Visible) then
            Write "1" into DVSF
        Else
            Write "0" into DVSF
        End if
        NoConflict = TRUE
        Go to beginning of User Schedule File
        Do while (not EOF for User Schedule File) and (NoConflict = TRUE)
            Get next record
            If [(tCALC >= tSTART_REC) and (tCALC <= tSTOP_REC)]
                Set NoConflict = FALSE
            End if
        End do while
        If (NoConflict) then
            Write "1" into Temp User Schedule File
        Else
            Write "0" into Temp User Schedule File
        End if
        t CALC = t CALC + 2sec
    End Do while
    Close USAT Interp EPH File, TDRS Interp EPH File, User Schedule File
    Go to beginning of DVSF and Temp User Schedule File
    Do while (not EOF of DVSF)
        Get next value from DVSF
        Get next value from Temp User Schedule File
        Multiply two values, write to DUASF
    End Do while
    Delete DVSF and Temp User Schedule File
    Close DUASF
End Procedure Generate DAR User Availability Schedule

```

#### 4.1.24 Procedure Calculate TDRS-USAT Visibility

Procedure Calculate TDRS-USAT Visibility

```
Define earth's radius as Re
tdrsAlt = sqrt (Xt^2 + Yt^2 + Zt^2) - Re
tdrsSlantToHorizon = sqrt (tdrsAlt * (2.0 * Re + tdrsAlt))
userAlt = sqrt (Xu^2 + Yu^2 + Zu^2) - Re
userSlantToHorizon = sqrt (userAlt * (2.0 * Re + userAlt))
sumOfSlants = tdrsSlantToHorizon + userSlantToHorizon
Xlos = Xt - Xu
Ylos = Yt - Yu
Zlos = Zt - Zu
DistanceBetweenSats = sqrt (Xlos^2 + Ylos^2 + Zlos^2)
If sumOfSlants >= DistanceBetweenSats
    Visible = TRUE
Else Visible = FALSE
End if
End Procedure Calculate TDRS-USAT Visibility
```

#### 4.1.25 Procedure Update DAR User Schedule

Procedure Update DAR User Schedule

// Reference Figure 6-2 and Table 6-1 of Process C

Added = FALSE

Deleted = FALSE

NoConflict = TRUE

ConflictError = FALSE

AddError = FALSE

DeleteError = FALSE

Get t<sub>START\_UPDATE</sub> and t<sub>STOP\_UPDATE</sub> for the update // from request specification

Get User Schedule OperationType // from request specification

Case (OperationType)

Schedule Add:

    Perform Add User Request

Schedule Delete:

    Perform Delete User Request

End case

// Error cases

If (OperationType = Schedule Add) and (NoConflict = FALSE) then Conflict = TRUE

If (OperationType = Schedule Add) and (Added = FALSE) then AddError = TRUE

If (OperationType = Schedule Delete) and (Deleted = FALSE) then DeleteError = TRUE

End Procedure Update DAR User Schedule

#### 4.1.26 Procedure Add User Request

Procedure Add User Request

```

Open User Schedule File // create new or open existing File
Open Temporary File
Perform Conflict Check // see if the add conflicts with existing request
If (NoConflict) then
    Do while (not EOF for User Schedule File)
        Read record from User Schedule File
        If (tREC > tNOW) // keep only future schedule entires
            If (tSTART_UPDATE < tSTART_REC) then
                Write new record to Temporary File
                Do while (not EOF for User Schedule File)
                    Write User Schedule File record to Temporary File
                    Read record from User Schedule File
                End Do while
                Added = TRUE
            End if
        Else
            Write User Schedule File record to Temporary File
        End else
    End if
    End do while
End if
If (EOF for User Schedule File) and (Add = FALSE)
    Append new record to Temporary File
    Added = TRUE
End if
Close User Schedule File
Close Temporary File
Copy Temporary File to User Schedule File
Delete Temporary File
End Procedure Add User Request

```

#### 4.1.27 Procedure Conflict Check

Procedure Conflict Check

```

Do while (not EOF for User Schedule File)
  Get record
  If [(tSTART_UPDATE >= tSTART_REC) and (tSTART_UPDATE <= tSTOP_REC)] or [(tSTOP_UPDATE >=
    tSTART_REC) and (tSTOP_UPDATE <= tSTOP_REC)]
    Set NoConflict = FALSE
  End if
End do while
End Procedure Conflict Check

```

#### 4.1.28 Procedure Delete User Request

Procedure Delete User Request

```

Open User Schedule File
Open Temporary File
Do while (not EOF for User Schedule File)
  Read record from User Schedule File
  If [(tSTART_UPDATE = tSTART_REC) and (tSTOP_UPDATE <= tSTOP_REC)] then
    Deleted = TRUE // skip over record to be deleted
  Else
    Write record from User Schedule File to Temporary File
  End if
End do while
Close User Schedule File
Close Temporary File
Copy Temporary File to User Schedule File
Delete Temporary File
End Procedure Delete User Request

```

#### 4.1.29 Procedure Check MAF Add User Request

Procedure Check MAF Add User Request

```

NoConflict = FALSE
Do while (not EOF for TUT file) and (NoConflict = FALSE)
  Get next TUT
  If [(tSTART_UPDATE >= tSTART_TUT) and (tSTOP_UPDATE <= tSTOP_TUT)]
    Set NoConflict = TRUE
  End if
End do while
End Procedure Check MAF Add User Request

```

#### 4.1.30 Procedure Build Message 111

Procedure Build Message 111  
Initialize Msg 111 buffer  
Insert User ID in buffer  
Insert TDRS ID in buffer  
Insert Service Start Time in buffer  
Insert Service Duration in buffer  
Insert Service Type in buffer  
Insert Service Support Type in buffer  
Insert Equipment ID in buffer  
If DAR service then  
    Insert DAR user type in buffer  
End if  
Insert Total Number of Time Slots in buffer  
Perform Availability Intervals  
Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 111 pointer  
End Procedure Build Message 111

#### 4.1.31 Procedure Build Message 112

Procedure Build Message 112  
Initialize Msg 112 buffer  
Insert User ID in buffer  
Insert TDRS ID in buffer  
Insert Service Start Time in buffer  
Insert Service Duration in buffer  
Insert Reference Action in buffer  
Case (ResultType) // **determined in the outside the procedure**  
    Granted:  
        Insert 00 into Result Code on buffer  
    Canceled:  
        Insert 01 into Result Code on buffer  
    Conflict:  
        Insert 02 into Result Code on buffer  
    Spooled:  
        Insert 03 into Result Code on buffer  
    Queued:  
        Insert 04 into Result Code on buffer  
End case  
Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 112 pointer  
End Procedure Build Message 112

#### 4.1.32 Procedure Build Message 113

Procedure Build Message 113  
  Initialize Msg 113 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  Insert Service Start Time in buffer  
  If MAF delete request then  
    Set delete type to MAF in buffer  
  Else  
    Set delete type to DAR in buffer  
  End if  
  If NoService then  
    Set Deletion Status bit in Msg buffer to Failed to Delete  
  Else  
    Set Deletion Status bit in buffer to Deleted  
  End if  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 113  
End Procedure Build Message 113

#### 4.1.33 Procedure Build Message 114

Procedure Build Message 114  
  Initialize Msg 114 buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 114 pointer  
End Procedure Build Message 114

#### 4.1.34 Procedure Build Message 115

Procedure Build Message 115  
  Initialize Msg 115 buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 115 pointer  
End Procedure Build Message 115

#### 4.1.35 Procedure Build Message 200

Procedure Build Message 200  
  Initialize Msg 200 buffer  
  Insert Change Start Time in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 200 pointer  
End Procedure Build Message 200

#### **4.1.36 Procedure Build Message 300**

Procedure Build Message 300  
  Initialize Msg 300 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  Insert Service Flag set to Delete in buffer  
  Insert Change Start Time in buffer  
  Insert Duration in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 300 pointer  
End Procedure Build Message 300

#### **4.1.37 Procedure Build Message 400**

Procedure Build Message 400  
  Initialize Msg 400 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  Insert Change ADD Start Time in buffer  
  Insert Duration in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 400 pointer  
End Procedure Build Message 400

#### **4.1.38 Procedure Build Message 401**

Procedure Build Message 401  
  Initialize Msg 401 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  Insert Change DELETE Start Time in buffer  
  Insert Duration in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 401 pointer  
End Procedure Build Message 401

#### 4.1.39 Procedure Build Message 500

Procedure Build Message 500  
  Initialize Msg 500 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  If Add Request then  
    Set ChangeType Flag to Add State  
  Else  
    Set ChangeType Flag to Delete State  
  End if  
  Insert MAF Change Start Time in buffer  
  Insert Duration in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 500 pointer  
End Procedure Build Message 500

#### 4.1.40 Procedure Build Message 501

Procedure Build Message 501  
  Initialize Msg 501 buffer  
  Insert User ID in buffer  
  Insert TDRS ID in buffer  
  If Add Request then  
    Set ChangeType Flag to Add State  
  Else  
    Set ChangeType Flag to Delete State  
  End if  
  Insert DAR Change Start Time in buffer  
  Insert Duration in buffer  
  Set TCP/IP socket Msg Buffer pointer to beginning of Msg Buffer 501 pointer  
End Procedure Build Message 501

#### 4.1.41 Procedure Availability Intervals

Procedure Availability Intervals

```

// for both MAF and DAR availability requests
  Open User Availability Schedule File // existing file
  Open Temporary Availability File // new file
  Last = 0
  StartAvailable = FALSE
  StopAvailable = FALSE
  IntervalCount = 0
  Do while (not EOF for User Availability Schedule File) // binary sched rep to time intervals
    Read record from User Availability Schedule File
    Get boolean data Next // 2nd field
    If [(Last = 0) and (Next = 1)] then // start transition
      Get time data Time // 1st field
      StartTime = Time
      Last = 0
      StartAvailable = TRUE
    End if
    If [(Last = 1) and (Next = 0)] then // stop transition
      Get time data Time // 1st field
      StopTime = Time
      Last = 1
      StopAvailable = True
    End if
    Last = Next
    If [(StartAvailable = TRUE) and (StopAvailable = TRUE)] then // avail interval exists
      Write StartTime and StopTime to Temporary Availability File
      StartAvailable = FALSE
      StopAvailable = FALSE
      Increment IntervalCount
    End if
  End do while
  If [(LAST = 1) and (StartAvailable = TRUE) and (StopAvailable = FALSE)] then // file endpt
    condition adjustment
    StopTime = Time
  End if
  Close User Availability Schedule File
  Initialize Message 111 Buffer
  If (IntervalCount > 0) // time is available
    Set file pointer to the beginning of the Temporary Availability File
    Do while (not EOF for Temporary Availability File)
      Read record from Temporary Availability File
      Get StartTime and StopTime
      Insert StartTime and StopTime in Message 111 Buffer
    End do while
    Insert IntervalCount in Message 111 buffer Total Number of Time Slots field
  Else // no time available

```

Insert 0 in Message 111 buffer Total Number of Time Slots field  
End if  
Close Temporary Availability File  
End Procedure Availability Intervals

## ***4.2 Process A Message Formats***

The following messages define the interface between the PT and Process A, and are defined in the Planning Tool PDL documentation:

Msg 101: PT to Process A, Available Time Request Message  
Msg 102: PT to Process A, Schedule Add Request Message  
Msg 103: PT to Process A, Schedule Delete Request Message  
Msg 111: Process A to PT, Available Time Message  
Msg 112: Process A to PT, Schedule Result Message  
Msg 113: Process A to PT, Schedule Deletion Notification Message  
Msg 114: Process A to PT, Forward Service Data Message  
Msg 115: Process A to PT, Return Service Data Message

The following message defines the interface between Process A and Process C, and is defined in the documentation for Process C:

Msg 300: Process A to Process C, Schedule Change

The subsections that follow present the interface definitions between Process A and the other DAP processes, as well as between Process A and the emulators:

#### 4.2.1 Process A Output Messages

**Table 4-1: Process A Output Messages**

MSG ID	Message Name	Destination	Description
200	TUT Change	Process B	Contains TUT change by MAF service addition or deletion
400	MAF Service Request	NCC	Request for normal MAF service within the TUT schedule
401	MAF Service Delete	NCC	Request to delete normal MAF service within the TUT schedule
500	DAB DAF Service Configuration	DAB Emulator	Configuration for DAF service data buffering
501	DAB DAR Service Configuration	DAB Emulator	Configuration for DAR service data buffering

#### 4.2.2 Process A Input Messages

**Table 4-2: Process A Input Messages**

MSG ID	Message Name	Source	Description
402	MAF Service Request Acknowledgement	NCC	Acknowledgement by NCC indicating MAF service has been implemented
403	MAF Service Delete Acknowledgement	NCC	Acknowledgement by NCC indicating MAF service has been deleted

### 4.2.3 Process A Data Outputs

**Table 4-3: Process A Data Outputs**

MSG ID	Data Item Name	Destination	Description
200	TUT Change start time	Process B	Start time for TUT change (YYDDDHMMSS)
200	TDRS ID	Process B	AAA (three alphanumeric characters)
200	User ID	Process B	User ID (four ASCII characters)
200	Change Type Flag	Process B	Flag (0 = addition of service, 1 = deletion)
200	Duration	Process B	Duration of TUT change (HHMMSS)
400	TDRS ID	NCC	AAA (three alphanumeric characters)
400	User ID	NCC	User ID (four ASCII characters)
400	Service Start Time	NCC	Start time for service (YYDDDHMMSS)
400	Duration	NCC	Duration of service (HHMMSS)
401	TDRS ID	NCC	AAA (three alphanumeric characters)
401	User ID	NCC	User ID (four ASCII characters)
401	Delete Start Time	NCC	Start time for service being deleted (YYDDDHMMSS)
401	Duration	NCC	Duration of service being deleted (HHMMSS)
500	TDRS ID	DAB Emulator	AAA (three alphanumeric characters)
500	User ID	DAB Emulator	User ID (four ASCII characters)
500	MAF Service Start Time	DAB Emulator	Start time for service (YYDDDHMMSS)
500	Duration	DAB Emulator	Duration of service (HHMMSS)
500	Change Type Flag	DAB Emulator	Flag (0 = addition of service, 1 = deletion)
501	TDRS ID	DAB Emulator	AAA (three alphanumeric characters)
501	User ID	DAB Emulator	User ID (four ASCII characters)
501	DAR Service Start Time	DAB Emulator	Start time for service (YYDDDHMMSS)
501	Duration	DAB Emulator	Duration of service (HHMMSS)
501	Change Type Flag	DAB Emulator	Flag (0 = addition of service, 1 = deletion)

#### 4.2.4 Process A Data Inputs

**Table 4-4: Process A Data Inputs**

<b>MSG ID</b>	<b>Data Item Name</b>	<b>Source</b>	<b>Description</b>
402	TDRS ID	NCC	AAA (three alphanumeric characters)
402	User ID	NCC	User ID (four ASCII characters)
402	Service Start Time	NCC	Start time for service (YYDDDHMMSS)
402	Duration	NCC	Duration of service (HHMMSS)
403	TDRS ID	NCC	AAA (three alphanumeric characters)
403	User ID	NCC	User ID (four ASCII characters)
403	Delete Start Time	NCC	Start time for service being deleted (YYDDDHMMSS)
403	Duration	NCC	Duration of service being deleted (HHMMSS)

### **4.3 Process A, B, and C Shared Files and Process A Internal Files**

#### **4.3.1 MAF TUT FILE RECORD FIELDS**

**Table 4-5: MAF TUT FILE RECORD FIELDS**

(Applies to Process A Procedures)

- Generate MAF User Availability Schedule
  - Verify MAF Service Exists)

Field Number	Field Name	Description
1	Start Time	TUT Interval Start Time
2	Stop Time	TUT Interval Stop Time

#### **4.3.2 TDRS INTERP EPHEMERIS FILE RECORD FIELDS**

**Table 4-6: TDRS INTERP EPHEMERIS FILE RECORD FIELDS**

(Applies to Process A Procedures:

- Generate DAR User Availability Schedule
- Generate MAF User Availability Schedule)

Field Number	Field Name	Description
1	Time	Interpolated Ephemeris Time
2	X	Interpolated X Position
3	Y	Interpolated Y Position
4	Z	Interpolated Z Position
5	ALPHA	Interpolated Pitch
6	BETA	Interpolated Roll
7	GAMMA	Interpolated Yaw

#### **4.3.3 TEMPORARY TUT FILE RECORD FIELDS**

**Table 4-7: TEMPORARY TUT FILE RECORD FIELDS**

(Applies to Process A Procedure Generate MAF User Availability Schedule)

Field Number	Field Name	Description
1	Time	Observation Time
2	Visibility Flag	Boolean Observation Result (Visible = 1/Not Visible = 0)

#### 4.3.4 TEMPORARY AVAILABILITY FILE RECORD FIELDS

**Table 4-8: TEMPORARY AVAILABILITY FILE RECORD FIELDS**  
**(Applies to Process A Procedure Availability Schedule)**

Field Number	Field Name	Description
1	Start Time	Schedule Interval Start Time
2	Stop Time	Schedule Interval Stop Time

#### 4.3.5 TEMPORARY DIRECTION COSINE FILE RECORD FIELDS

**Table 4-9: TEMPORARY DIRECTION COSINE FILE RECORD FIELDS**

Field Number	Field Name	Description
1	Time	Direction Cosine Evaluation Time
2	$\alpha$	See section 5.3 of TGBFS ICD Message 3 Format
3	$\beta$	"
4	$\gamma$	"

#### 4.3.6 TEMPORARY DIRECTORY FILE RECORD FIELDS

**Table 4-10: TEMPORARY DIRECTORY FILE RECORD FIELDS**  
**(Applies to Process A Procedures:**

- Determine Directories
- Delete TGBFS Messages
  - Load DARDB
  - Initialize DARDB

Field Number	Field Name	Description
1	Start Time	Schedule Interval Start Time
2	Stop Time	Schedule Interval Stop Time

#### 4.3.7 TEMPORARY MESSAGE FILE RECORD FIELDS

**Table 4-11: TEMPORARY MESSAGE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Load DARDB Request)**  
**[See UserID File Table in Process C for Format of 808 Byte Records]**

### 4.3.8 TEMPORARY USER FILE RECORD FIELDS

**Table 4-12: TEMPORARY USER FILE RECORD FIELDS**  
**(Applies to Process A Procedure Delete TGBFS Messages Request)**  
**[See UserID File Table in Process C for Format of 808 Byte Records]**

### 4.3.9 TEMPORARY USERID FILE RECORD FIELDS

**Table 4-13: TEMPORARY USERID FILE RECORD FIELDS**  
**(Applies to Process A Procedure Insert Messages into UserID File)**  
**[See UserID File Table in Process C for Format of 808 Byte Records]**

### 4.3.10 USAT INTERP EPH FILE RECORD FIELDS

**Table 4-14: USAT INTERP EPH FILE RECORD FIELDS**

**(Applies to Process A Procedures:**

- Generate MAF User Availability File
- Generate DAR User Availability Schedule
- Generate TDRS-USAT Visibility Schedule

**Applies to Process B Procedure:**

- Create Interp EPH File)

Field Number	Field Name	Description
1	Time	Interpolated Ephemeris Time
2	X	Interpolated X Position
3	Y	Interpolated Y Position
4	Z	Interpolated Z Position
5	ALPHA	Interpolated Pitch
6	BETA	Interpolated Roll
7	GAMMA	Interpolated Yaw

### 4.3.11 MAF USER AVAILABILITY SCHEDULE FILE RECORD FIELDS

**Table 4-15: MAF USER AVAILABILITY SCHEDULE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Generate MAF Visibility Schedule)**

Field Number	Field Name	Description
1	Time	Observation Time
2	Visibility Flag	Boolean Observation Result (Visible = 1/Not Visible = 0)

#### 4.3.12 DAR USER AVAILABILITY SCHEDULE FILE RECORD FIELDS

**Table 4-16: DAR USER AVAILABILITY SCHEDULE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Generate DAR Visibility Schedule)**

Field Number	Field Name	Description
1	Time	Observation Time
2	Visibility Flag	Boolean Observation Result (Visible = 1/Not Visible = 0)

#### 4.3.13 MAF USER SCHEDULE FILE RECORD FIELDS

**Table 4-17: MAF USER SCHEDULE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Generate MAF Visibility Schedule)**

Field Number	Field Name	Description
1	Start Time	Schedule Interval Start Time
2	Stop Time	Schedule Interval Start Time

#### 4.3.14 DAR USER SCHEDULE FILE RECORD FIELDS

**Table 4-18: DAR USER SCHEDULE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Generate DAR Visibility Schedule)**

Field Number	Field Name	Description
1	Start Time	Schedule Interval Start Time
2	Stop Time	Schedule Interval Start Time

#### 4.3.15 VISIBILITY SCHEDULE FILE RECORD FIELDS

**Table 4-19: VISIBILITY SCHEDULE FILE RECORD FIELDS**  
**(Applies to Process A Procedure Generate DAR Visibility Schedule)**

Field Number	Field Name	Description
1	Time	Observation Time
2	Visibility Flag	Boolean Observation Result (Visible = 1/Not Visible = 0)

### 4.3.16 TGBFS Messages for Phase 1 Scenario

**Table 4-20: TGBFS Messages for Phase 1 Scenario**

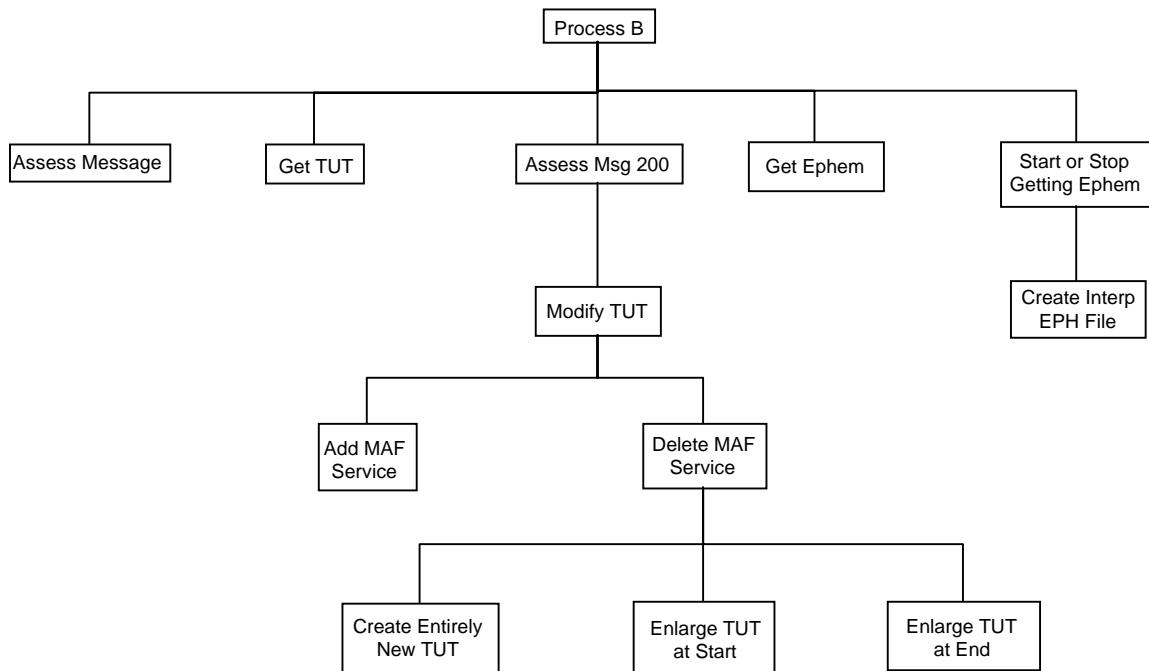
INITIALIZATION	RUN TIME	SHUTDOWN
EMC Control Data Message Message Type = 0 Calibration Mode = {External, internal} Reset Option ={Yes, No} Clock Select ={Local, Prim, Sec) HDLC Source = {Prim,Redun. } EMC Mode = On Line Operator Specified Element Status	User Dir Cosines Data Message (IF A Pointing Mode) Message Type = 3  Which IBU {0,1,2,3,4,5} Alpha  Beta Gamma	
IBUG Control Data Message Message Type = 4 Which IBU ={0,1,2,3,4,5} Reset Option FCRX Failover Control FCRX Select ={Prim, Redun. } IBU Mode IBUG Mode = On Line		(NONE)
User Dir Cosines Data Message (IF A Pointing Mode) Message Type = 3 Which IBU ={0,1,2,3,4,5} Alpha Beta Gamma		

## 5. DAP Process B DAB Data Base Manager Detailed Design

The following assumptions are made for Process B functions during Phase 1 implementation:

- Process B is divided into 2 distinct parts. The first part loads the TUT from the NCC and the ephemerides from the FDF, converts the message data into internal DAP format, and stores them in the TUT DB and EPHEM DB. The second part accepts TUT Change messages from Process A, and implements the TUT changes
- Process B cannot accept messages from Process A until the NCC and the FDF have sent TUTs and ephemerides at least once
- For Phase 1, only one TDRS and one USAT ephemerides set will be loaded, extending over the period of a day, and these ephemerides will have a 10 second time step
- The TUT schedule loaded from the NCC will apply to a single TDRSS satellite MAF link. It will consist of the start and stop times of each TDRS Unused Time
- The MAF schedule adds and deletes from Process A. Messages from A are required indicating that an add or delete is granted, then B modifies the TUT to reflect this change
- Process A checks to see if a service exists, in the case of a delete, Process A checks to see if a service time is available, in the case of an add, so Process B does not need to perform these checks.

Figure 5-1 shows the structure chart for functions of Process B during Phase 1 implementation.



**Figure 5-1: DAP Phase 1 Procedure hierarchies for Process B**

## 5.1 PDL Procedures for Process B

The following subsections describe the PDL for Process B functions during Phase 1 implementation.

### 5.1.1 Begin Process B

```
Begin Process B
    Initialize process // distinct from Process B1 or B2
    TUTRecv = FALSE // No TUT file in existence yet
    EPHRecv = FALSE // No ephemeris file in existence yet
    // Endless main processing loop
    Do while TRUE
        // Wait for new Msg
        Wait for message
        If message arrives
            Get Msg ID
            Perform Procedure Assess Message
            // Take the appropriate action
            Case (ID)
                200: Procedure Assess Msg 200
                600: Procedure Get TUT
                700: Procedure Get Ephem
                701: Procedure Start or Stop Getting Ephem
                2000: Print out error message
            End case
        End if // message arrives
    End do while
End Process B
```

### 5.1.2 Procedure Assess Message

Procedure Assess Message

    ID = 2000 // **Error Code**

    If Msg ID = 200

        ID = 200

    End if

    Else if Msg ID = 600

        ID = 600

    End if

    Else if Msg ID = 700

        ID = 700

    End if

    Else if Msg ID = 701

        ID = 701

    End if

End Procedure Assess Message

### 5.1.3 Procedure Assess Msg 200

```
Procedure Assess Msg 200
  If (TUTRecv = FALSE) or (EPHRecv = FALSE)
    If (TUTRecv = FALSE)
      Print out "TUT not yet received"
    End if
    If (EPHRecv = FALSE)
      Print out "Ephemeris not yet received"
    End if
  Else Procedure Modify TUT
  End if
End Procedure Assess Msg 200
```

### 5.1.4 Procedure Modify TUT

```
Procedure Modify TUT
  Get t_start_update // from request specification
  Get duration "d" // from request specification
  t_stop_update = t_start_update + d
  Case (Change Type Flag)
    Service Addition:
      Procedure Add MAF Service
    Service Deletion:
      Procedure Delete MAF Service
  End case
End Procedure Modify TUT
```

### 5.1.5 Procedure Add MAF Service

```
Procedure Add MAF Service
    Open MAF TUT File
    Open Temporary File
    Do while (not EOF for MAF TUT File)
        Read TUT record from the MAF TUT File
        If (tstart_update >= tstart_tut) and (tstop_update <= tstop_tut)
            If (tstart_update > tstart_tut)
                Create new TUT with tstart_tut_new = tstart_tut
                tstop_tut_new = tstart_update
                Write new TUT record to Temporary File
            End if
            If (tstop_update < tstop_tut)
                Create new TUT with tstart_tut_new = tstop_update
                tstop_tut_new = tstop_tut
                Write new TUT record to Temporary File
            End if
        End if
        Else
            Write TUT record to Temporary File
        End else
    End do while
    Close MAF TUT File
    Close Temporary File
    Copy Temporary File to MAF TUT File
    Delete Temporary File
End Procedure Add MAF Service
```

### 5.1.6 Procedure Delete MAF Service

```
Procedure Delete MAF Service
    Open MAF TUT File
    Open Temporary File
    Deleted = FALSE
    Do while (not EOF for MAF TUT File)
        Read TUT record from the MAF TUT File
        If (tstop_tut >= tstart_update)
            If (tstart_tut > tstop_update)
                Procedure Create Entirely New TUT
                Write existing TUT record to Temporary File
                Deleted = TRUE
            End if
            Else if (tstart_tut = tstop_update)
                Procedure Enlarge TUT at Start
                Deleted = TRUE
            End if
            Else if (tstop_tut = tstart_update)
                Procedure Enlarge TUT at End
                Deleted = TRUE
            End if
        End if
        Else
            Write TUT record to Temporary File
        End else
    End do while
    If (EOF for MAF TUT File) and (Deleted = FALSE)
        Procedure Create Entirely New TUT
        Deleted = TRUE
    End if
    Close MAF TUT File
    Close Temporary File
    Copy Temporary File to MAF TUT File
    Delete Temporary File
End Procedure Delete MAF Service
```

### 5.1.7 Procedure Create Entirely New TUT

```

Procedure Create Entirely New TUT
  Create new TUT with tstart_tut_new = tstart_update
  tstop_tut_new = tstop_update
  Write new TUT record to Temporary File
End Procedure Create Entirely New TUT

```

### 5.1.8 Procedure Enlarge TUT at Start

```

Procedure Enlarge TUT at Start
  Create new TUT with tstart_tut_new = tstart_update
  tstop_tut_new = tstop_tut
  Write new TUT record to Temporary File
End Procedure Enlarge TUT at Start

```

### 5.1.9 Procedure Enlarge TUT at End

```

Procedure Enlarge TUT at End
  Create new MAF TUT with tstart_tut_new = tstart_tut
  If (not EOF for MAF TUT File)
    Read TUT record from the MAF TUT File
    If (tstart_tut = tstop_update)
      tstop_tut_new = tstop_tut
      Write new TUT record to Temporary File
    Else
      tstop_tut_new = tstop_update
      Write new TUT record to Temporary File
      Write existing TUT record to Temporary File
    End if
  Else
    tstop_tut_new = tstop_update
    Write new TUT record to Temporary File
  End if
End Procedure Enlarge TUT at End

```

### **5.1.10 Procedure Get TUT**

```

Procedure Get TUT
    Open New MAF TUT File
    Do while (not at end of Msg 600)
        Get start, stop time of next TUT from Msg 600 fields
        Write start, stop time to new MAF TUT File
    End do while
    Close new TUT File
    If (TUTRecv = FALSE)
        TUTRecv = TRUE
    Else replace old MAF TUT File with new MAF TUT File
    End if
End Procedure Get TUT

```

### **5.1.11 Procedure Get Ephem**

```

Procedure Get Ephem
    Get t, x, y, z,  $\alpha$ ,  $\beta$ ,  $\gamma$  from Msg 700
    Write t, x, y, z,  $\alpha$ ,  $\beta$ ,  $\gamma$  to EPH File
End Procedure Get Ephem

```

### **5.1.12 Procedure Start or Stop Getting Ephem**

```

Procedure Start or Stop Getting Ephem
    Read Action Flag from Msg 701
    If Action Flag = "Start"
        Determine which ephemeris file to open
        Open EPH File
    End if
    Else if Action Flag = "Stop"
        Close modified EPH File
        EPHRecv = TRUE
        Procedure Create Interp EPH File
    End if
End Procedure Start or Stop Getting Ephem

```

### 5.1.13 Procedure Create Interp EPH File

Procedure Create Interp EPH File

    Open EPH File

    Open Interp EPH File

    Get 1st record from EPH File

    Set t2, x2, y2, z2 from record

    Get  $\alpha$ ,  $\beta$ ,  $\gamma$  from record

    Repeat

        Set t1 = t2, x1 = x2, y1 = y2, and z1 = z2

        Get next record from EPH File

        Set t2, x2, y2, z2 from record

        Set t = t1

        While (t < t2)

$x = x1 + ((x2 - x1) / (t2 - t1)) * (t - t1)$

$y = y1 + ((y2 - y1) / (t2 - t1)) * (t - t1)$

$z = z1 + ((z2 - z1) / (t2 - t1)) * (t - t1)$

            Write t, x, y, and z to Interp EPH File

            Write  $\alpha$ ,  $\beta$ ,  $\gamma$  to Interp EPH File

$t = t + 2 \text{ sec}$

        End while

    Until (EOF of EPH File)

    Write t2, x2, y2, and z2 to Interp EPH File

    Write  $\alpha$ ,  $\beta$ ,  $\gamma$  to Interp EPH File

    Close EPH File

    Close Interp EPH File

End Procedure Create Interp EPH File

## 5.2 Process B Message Formats

Message 200, sent from Process A to Process B, has been defined in the Process A documentation

### 5.2.1 Process B Input Messages

**Table 5-1: Process B Input Messages**

MSG ID	Message Name	Source	Description
600	TUT From NCC	NCC	Contains start and stop times for MAF TUT for one TDRS
700	Ephemeris at one time	FDF	Contains position, attitude information for one satellite at one time
701	Ephemeris start/stop alert	FDF	An alert message which tells B that ephemeris messages (700) are going to be sent, or that they have all been sent

### 5.2.2 Process B Data Inputs

**Table 5-2: Process B Data Inputs**

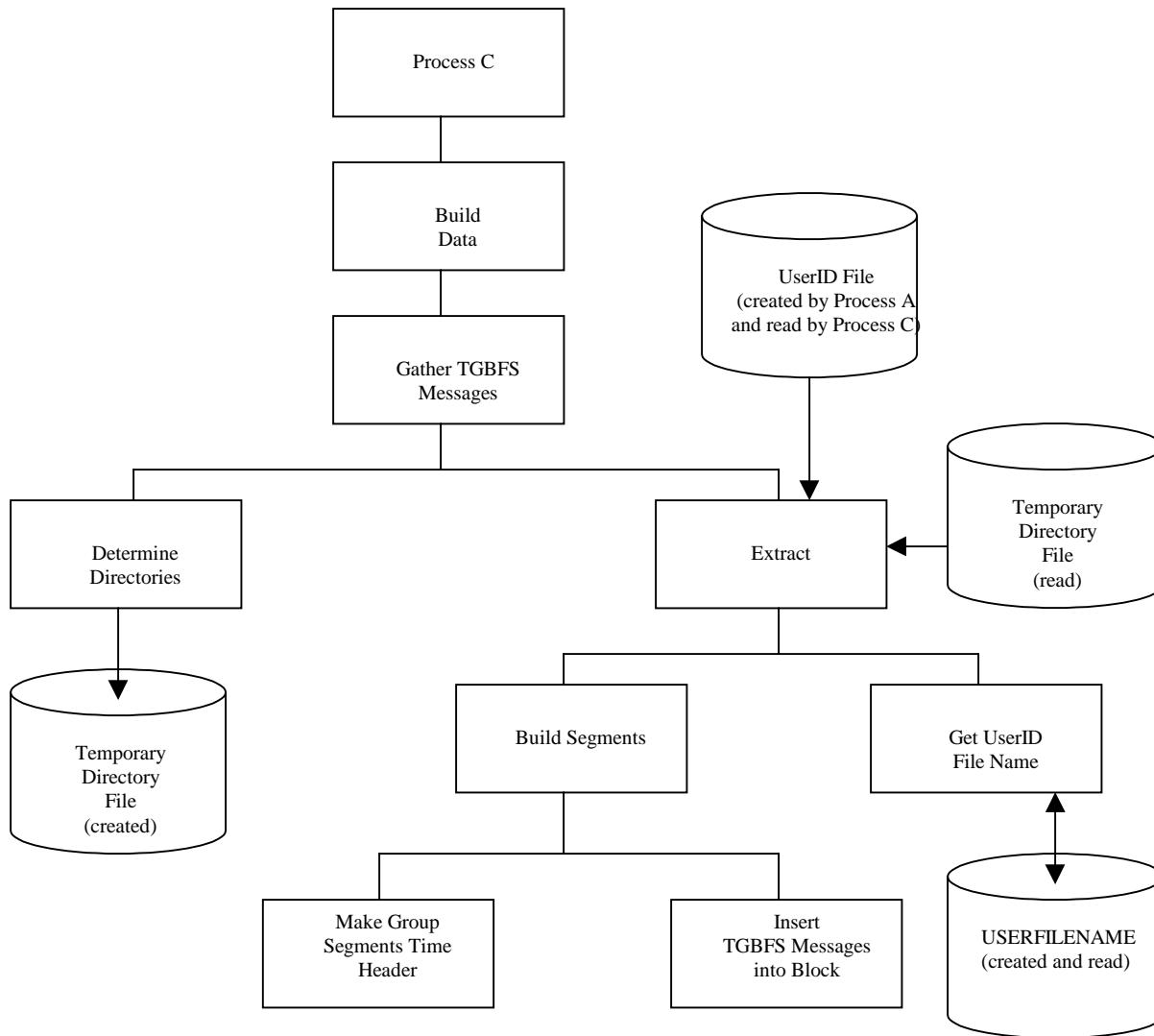
MSG ID	Data Input Name	Source	Description
600	TDRS ID	NCC	AAA (three alphanumeric characters)
600	TUT Start Time-1	NCC	YYDDDHMMSS
600	TUT Stop Time-1	NCC	YYDDDHMMSS
600	..... (other slots, as in previous 2 fields)	NCC	.....
600	End of TUT	NCC	End of file flag
700	Satellite ID	FDF	Code designating satellite number
700	Time	FDF	Time of the ephemeris entry YYDDDHMMSS
700	X	FDF	X coordinate (km)
700	Y	FDF	Y coordinate (km)
700	Z	FDF	Z coordinate (km)
700	Alpha	FDF	Pitch (degrees)
700	Beta	FDF	Roll (degrees)
700	Gamma	FDF	Yaw (degrees)
701	Ephemeris start/stop alert flag	FDF	Flag which indicates ephemeris records are starting to arrive, or have all been sent (0/1)

## 6. DAP Process C TGBFS Data Preparation Detailed Design

The following assumptions are made for Process C functions during Phase 1 implementation:

- Process A maintains the DAR DB (Figures 6-2, 6-3, and Tables 6-1 and 6-2) and Process C reads it
- Process A notifies Process C when a change has been made to the DAR DB
- Process C uses non-adaptive tunable parameters
- Process C allows one change in the DAR DB to influence a constructed data block waiting for transmission to Process D (Phase 2 implementation)
- Each data block generated by C contains at least several minutes of data to be transmitted in real time by Process D
- Process C prepares data block with segment components of 2 sec time steps (TGBFS ICD Msg 3 rate requirement direction cosine update requirement for each DAP user)
- At a minimum, the data block group segment contains one or more TGBFS Msg 3 data structures for one or more DAP DAR users, respectively
- Process C is driven by the need to gather large quantities the TGBFS data well in advance of the send time and label it with the real-time that it must be sent to the TGBFS by Process D
- Process C takes raw TGBFS data prepared by Process A (based on the DAP user schedules), blocks it into large packages, and ships it to Process D that must send the data at regular 2 second intervals (See Figure 6-7 for relative Process processing timing requirements).

Figure 6-1 shows the structure chart for functions of Process C during Phase 1 implementation.

**DRAFT****Version 1.0**

**Figure 6-1: DAP Phase 1 Procedure hierarchies for Process C**

**Table 6-1 - DAP User Schedule in DAR DB**

USER	TDRS	START TIME	STOP TIME
A	171	13:00	13:30
B	171	13:00	13:15
C	171	13:20	13:30
A	171	13:45	14:00
B	171	13:45	14:15
C	171	14:30	15:00
D	171	14:45	15:15

**Table 6-2 - Stored TGBFS Data Format in DAR DB  
(See Table 6-7 for File Record Layout)**

DIR: 1300	FILE: A	FILE: B	FILE: C	
	0000 data	0000 data	1200 data	
	0002 data	0002 data	1202 data	
	.....	.....	.....	
	1798 data	0900 data	1798 data	
DIR: 1330	FILE: A	FILE: B	FILE: C	
	0000 data	0900 data	0000 data	
	0002 data	0902 data		
	.....	.....		
	0900 data	0900 data		
DIR: 1400		FILE: B		
		0002 data		
		0004 data		
		....		
		0900 data		
DIR: 1430				FILE: D
				0900 data
				0902 data
				.....
				1798 data
DIR: 1500				FILE: D
				0002 data
				0004 data
				....
				0900 data

## 6.1 PDL Procedures for Process C

The following subsections describe the PDL for Process C functions during Phase 1 implementation.

### 6.1.1 Begin Process C

```

Begin Process C
    // TGBFS Data Block Generation and Send Processing

    Initialize process C
    NoMsgData ← TRUE // first time through data block build process
    // TWINDOW is a tunable constant (function of number of users and min schedule time)
    Set TWINDOW ← N minutes
    // TBUILD_THRESH is a tunable constant (max time to build a data block)
    Set TBUILD_THRESH ← W minutes // TBUILD_THRESH ≤ (TWINDOW)/2
    // initialize send time to allow first block build to occur via Process A change message
    Set tSEND ← TWINDOW + tNOW
    Set tSTART ← tNOW

    Do while TRUE // endless processing loop
        Wait for time out at tSEND or change notification message 300 from Process A
        If a schedule change notification message comes from Process A // see Figure 6-5
            Get the change start time tc // tc ≥ tNOW
            If (tc ≤ tSTART + TBUILD_THRESH) then // enough time to rebuild datablock
                If (NoMsgData) then
                    NoMsgData ← FALSE
                Else // block has already been built
                    Discard just built data block
                End if
                Perform Build_Block
            End if
            Else // tSEND time out has occurred (not a Process A change notification)
                If (not NoMsgData) then
                    Send data block to Process D via message 1000
                    Perform Build_Block
                End if
            End if
        End do while // endless processing loop
    End Process C // TGBFS Data Block Generation and Send Processing

```

### 6.1.2 Procedure Build\_Block

```

Procedure Build_Block
// Builds Data Block from data in DAR DB
    Set tSEND ← tSTART + TWINDOW
    Check DAR DB schedule for users in next time window // see Figure 6-3 and Table 6-1
    If no data // no users scheduled in the next time window then
        NoMsgData ← TRUE
    Else
        NoMsgData ← FALSE
        Perform Gather TGBFS Data
        Set tSTART ← tSEND // get ready for the next data block
    End if
End Procedure Build_Block // Builds Data Block from data in DAR DB

```

### 6.1.3 Procedure Gather TGBFS Msg Data

```

Procedure Gather TGBFS Msg Data
    Perform Determine Directories // find the DAR DB directories containing the data to be
                                blocked
    Perform Extract Msgs
End Procedure Gather TGBFS Msg Data

```

### 6.1.4 Procedure Determine Directories

```

Procedure Determine Directories
// THIS PROCEDURE IS IDENTICAL TO THAT FOUND IN PROCESS A
    Open Temporary Directory File // see Table 6-9
    t = t START
    Do while (t <= t SEND )
        Convert t to ASCII string T
        Truncate minute and seconds bytes from T // implementation note: last 4 bytes
        Write T to the next Temporary Directory File Record // these are the directory names
        t = t + 1 hour
    End do while
    Close Temporary Directory File
End Procedure Determine Directories

```

### 6.1.5 Procedure Extract Msgs

Procedure Extract Msgs

Open Temporary Directory File // **Temporary Directory File ≡ TDF**

Initialize data block buffer pointer DBBP to the beginning of the data block being built

Do while (not EOF for TDF)

    Read record TDFREC form TDF

    Get name T from TDFREC

    Perform Get UserID File Name // **User File ≡ UF**

    Open UserID File // **implementation note: path is /DARDB/T/UFN**

    Do while (not EOF for UF)

        Read record UFREC from UF

        Get t<sub>UFREC</sub> from UFREC

        If [(t<sub>UFREC</sub> >= t<sub>START</sub>) and (t<sub>UFREC</sub> <= t<sub>SEND</sub>)] then

            Perform Build Segments

        End if

    End do while

    Close UserID File

End do while

Close Temporary Directory File

End Procedure Extract Msgs

### 6.1.6 Procedure Get UserID File Name

Procedure Get UserID File Name

    system (rm USERFILENAME) // **actual C code except for the correct path**

    Build string "ls /DARDB/T > USERFILENAME"

    system ("ls /DARDB/T > USERFILENAME") // **actual C code except for the complete path**

    Open USERFILENAME // **need to prefix with correct path components (see Table 6-10)**

    Read record from USERFILENAME // **only one record possible in USERFILENAME for Phase 1**

    Get UserID File name UFN from Record

    Close USERFILENAME

End Procedure Get UserID File Name

### 6.1.7 Procedure Build Segments

Procedure Build Segments

    Read the MsgMap field of the UFREC // contains list of msgs to send at t<sub>UFREC</sub>

    Perform Make Group Segment Time Header

    Perform Insert TGBFS Messages into Block

End Procedure Build Segments

### 6.1.8 Procedure Make Group Segment Time Header

Procedure Make Group Segment Time Header

Count number of "on" bits in MsgMap to determine Number of TGBFS Messages NumMsgs in UFREC

Look up length of TGBFS messages for each MsgMap "on" bit and total to get TotalSize in bytes

Insert t<sub>UFREC</sub>, TotalSize, and NumMsgs the address DBBP + the required offsets for each header item

Increment the pointer DBBP to point to the start byte of the first TGBFS message

End Procedure Make Group Segment Time Header

### 6.1.9 Procedure Insert TGBFS Messages into Block

Procedure Insert TGBFS Messages into Block

If Bit Number 15 = 1 then // **TGBFS initialization service scenario**

Extract TGBFS Message 0 data from UFREC

Insert TGBFS Message 0 data into data block starting at DBBP

Increment DBBP by size of TGBFS Message 0

Extract TGBFS Message 4 data from UFREC

Insert TGBFS Message 4 data into data block starting at DBBP

Increment DBBP by size of TGBFS Message 4

Extract TGBFS Message 3 data from UFREC

Insert TGBFS Message 3 data into data block starting at DBBP

Increment DBBP by size of TGBFS Message 3

Else // **TGBFS post initialization service scenario**

Extract TGBFS Message 3 data from UFREC

Insert TGBFS Message 3 data into data block starting at DBBP

Increment DBBP by size of TGBFS Message 3

End Procedure Insert TGBFS Messages into Block

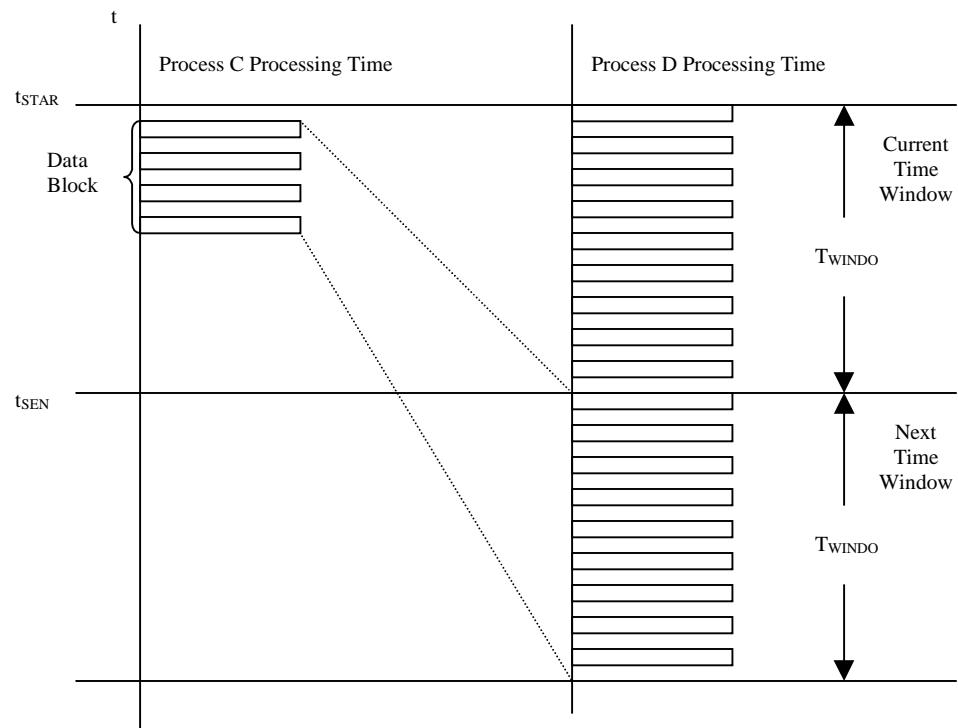


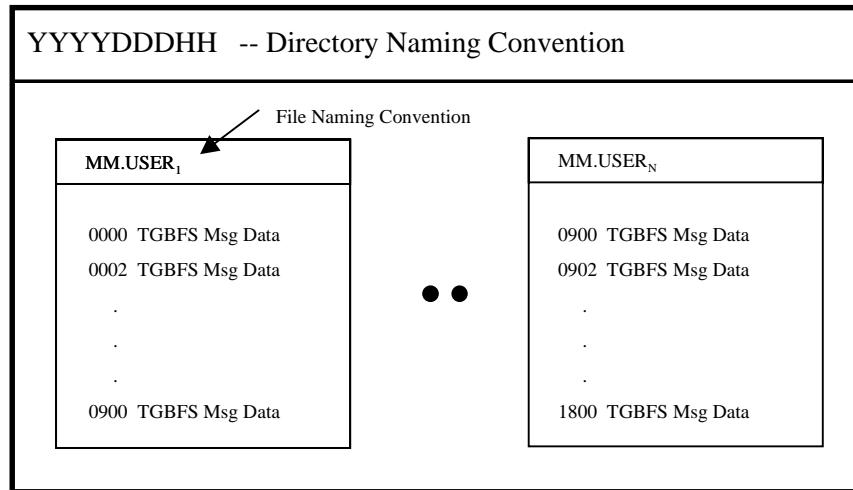
Figure 6-2: Simplest Data Block Timing Scenario without Schedule Change

<u>USER ID</u>	<u>START</u>	<u>STOP</u>	<u>TDRS</u>
A	199801221306	199801221316	171
B	199801221553	199801221630	171
C	199801221555	199801231600	171
A	199801230025	199801231645	171
	etc.		

**Figure 6-3: DAR User Schedule Contained in DAR DB**

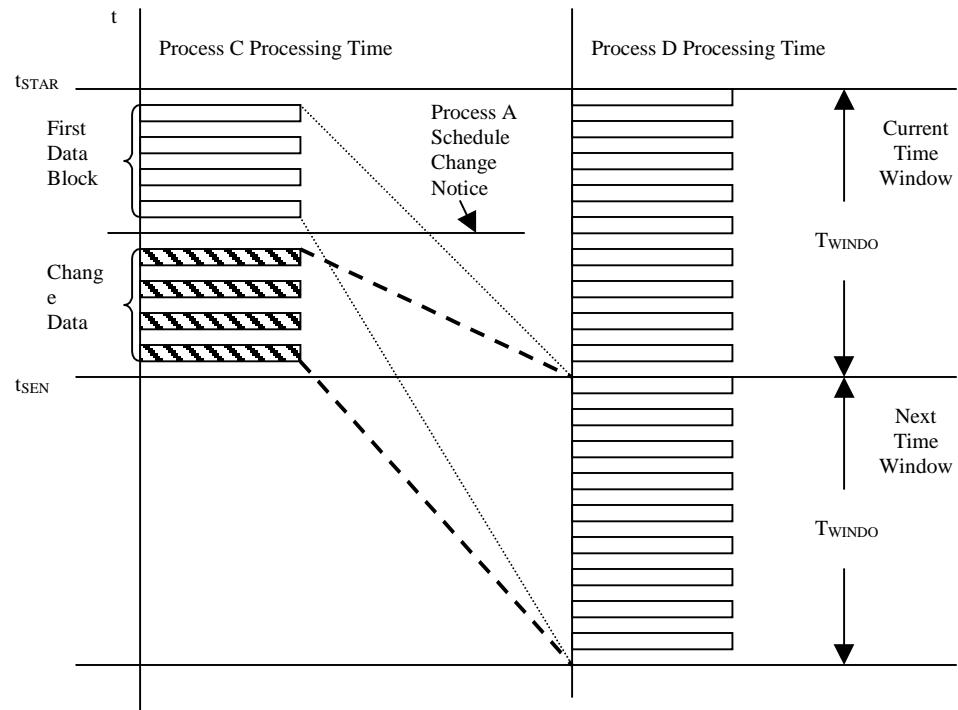
# DAR DB STRUCTURE TO MAINTAIN TGBFS MSG DATA BY TIME AND DAP USER

(Produced and Maintained by Process A and to Read by Process C)

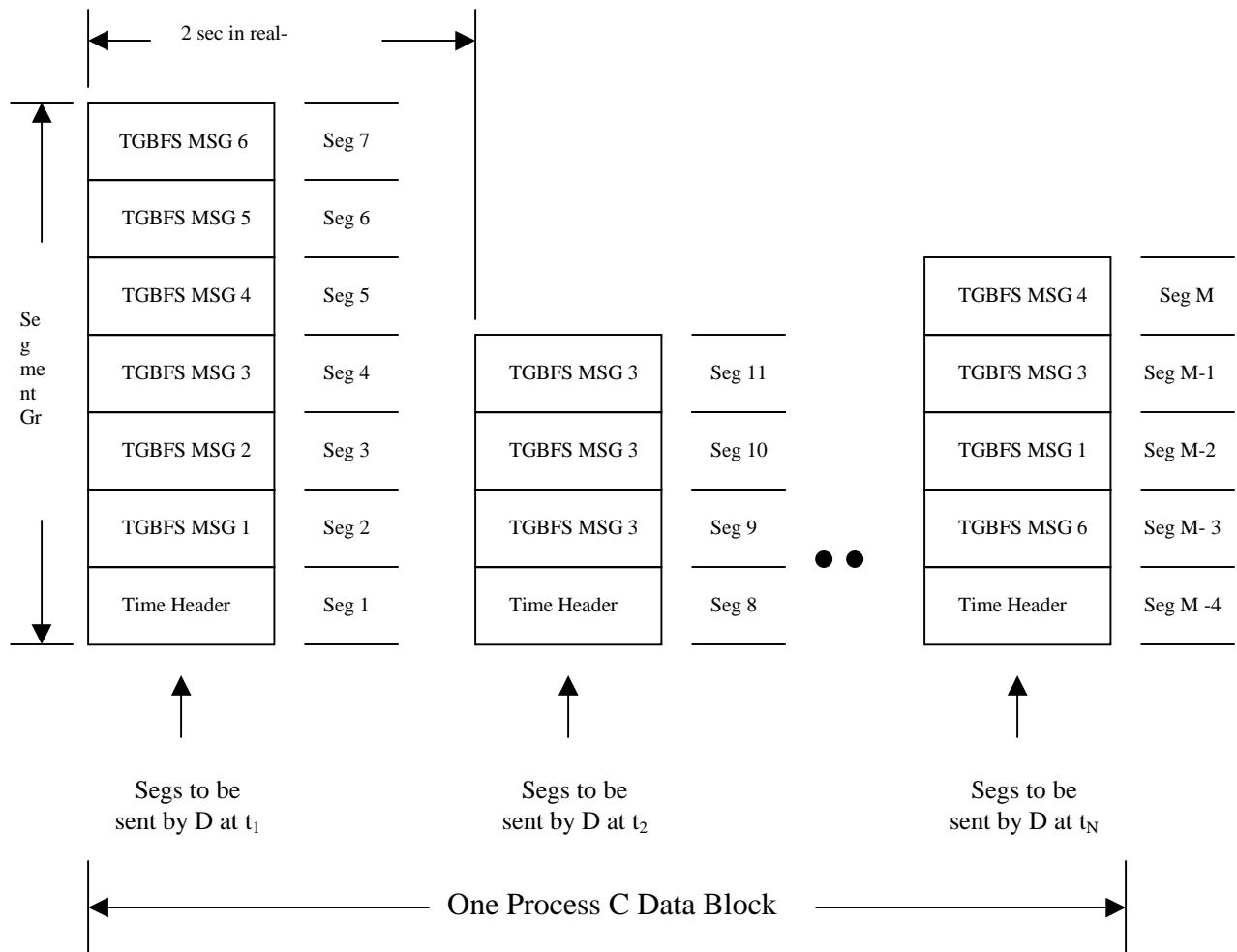


Storage Requirements: 43,200 Records/User/Day (Max)  
~ 1 kB per record (Max)

**Figure 6-4: DAR DB TGBFS Message Data Storage**



**Figure 6-5: Simplest Data Block Timing Scenario with Simplest Schedule Change**



**Figure 6-6: Sample Data Block Layout**

## **6.2 Process C Message Formats**

### **6.2.1 Process C Input Messages**

**Table 6-3: Process C Input Messages**

<b>MSG ID</b>	<b>Message Name</b>	<b>Source</b>	<b>Description</b>
300	Schedule Change	Process A	Contains DAR Schedule Time Change

### **6.2.2 Process C Data Inputs**

**Table 6-4: Process C Data Inputs**

<b>MSG ID</b>	<b>Data Item Name</b>	<b>Data Type</b>	<b>Description</b>
300	TimeChange	unsigned long integer	Start time for first change to the schedule in UNIX system time

### **6.2.3 Process C Output Messages**

**Table 6-5: Process C Output Messages**

<b>MSG ID</b>	<b>Message Name</b>	<b>Destination</b>	<b>Description</b>
1000	TGBFS Data Block	Process D	Contains a contiguous sequence of user interleaved TGBFS messages with send times to be sent by Process D to the TGBFS at 2 sec intervals

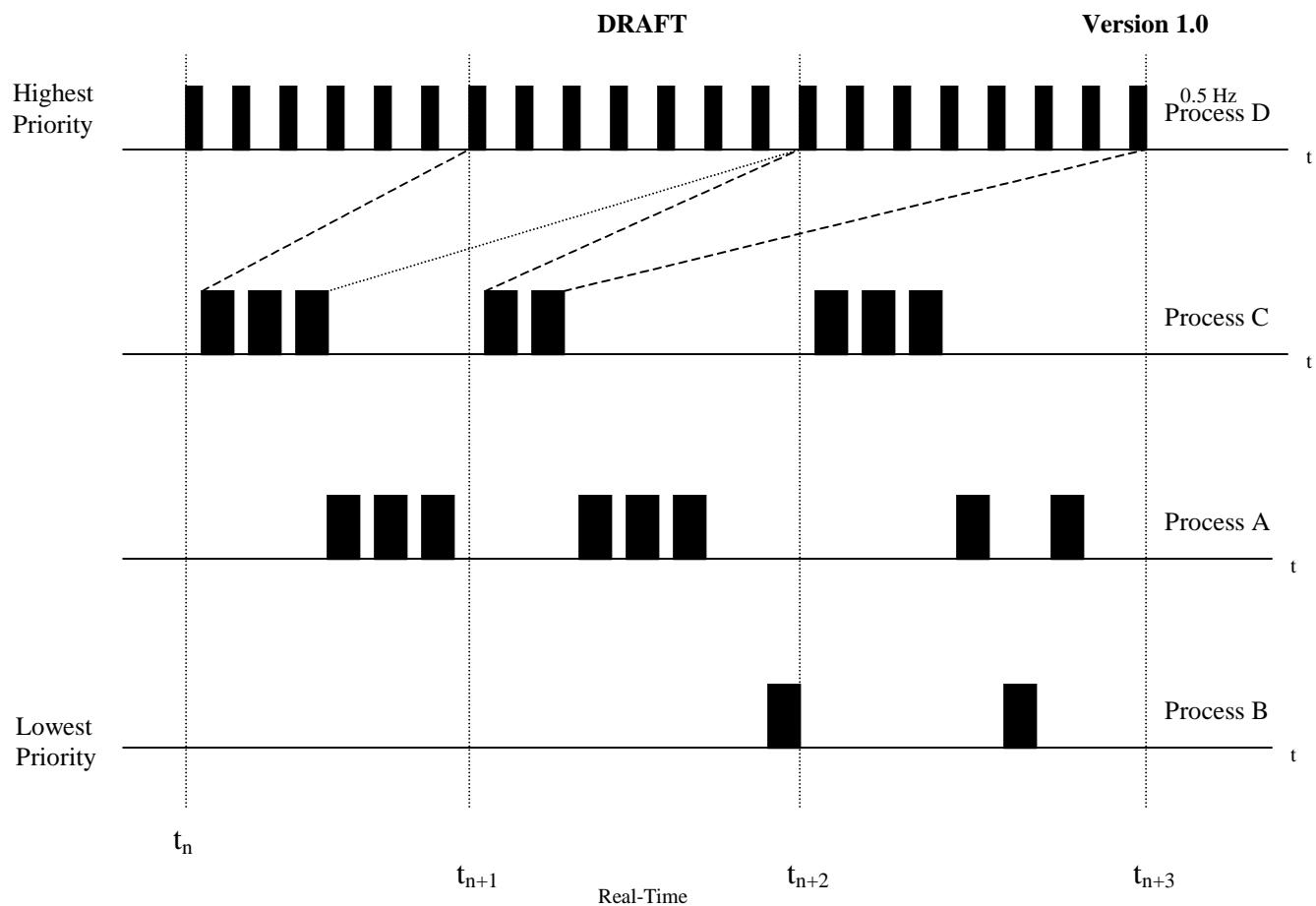
### 6.2.4 Process C Data Outputs

**Table 6-6: Process C Data Outputs**

MSG ID	Data Item Name	Source	Description
1000	TGBFS Msg (User_Direction_Cosines)	DAR DB	See TGBFS ICD Sec. 5.3.1 (20 Bytes of data <u>once every 2 sec</u> ) TGBFS Msg ID 3
1000	TGBFS Msg (EMC_Control_Data)	DAR DB	See TGBFS ICD Sec. 5.3.2 (12 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 0
1000	TGBFS Msg (IBUG_Control_Data)	DAR DB	See TGBFS ICD Sec. 5.3.3 (16 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 4
1000	TGBFS Msg (TDRS_Array_Geometry_Data)	DAR DB	See TGBFS ICD Sec. 5.3.4 (244 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 5
1000	TGBFS Msg (Element_Data)	DAR DB	See TGBFS ICD Sec. 5.3.5 (248 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 2
1000	TGBFS Msg (Fixed_Weight_Data)	DAR DB	See TGBFS ICD Sec. 5.3.6 (248 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 6
1000	TGBFS Msg (Report_Request_Message)	DAR DB	See TGBFS ICD Sec. 5.3.7 (12 Bytes of data at greater than 2 sec intervals) TGBFS Msg ID 1
1000	Time Header	Computed by C	See Table 6-8 for the header layout

### ***6.3 Timing Requirements Among DAP Processes***

- Process D strips segment groups for Process C's data blocks and sends control messages to the TGBFS Emulator at a fixed 0.5 Hz rate
- Process C builds in advance the user interleaved TGBFS messages for each 2 second time interval (e.g., in a several minute period) and assembles them into one large data block for TCP/IP message 10 transfer to Process D
- Process A
  1. handles DAP user request for DAF and DAR information and service
  2. notifies the NCC Emulator when a DAF request has been approved
  3. notifies the DAB Emulator when a DAF or DAR service has been approved
  4. maintains the DAR DB with user schedules and constructs raw TGBFS message data and stores it for each DAR service request
  5. notifies Process C each time a schedule change occurs via message 1
- Process B updates the DAF DB when NCC and FDF Emulator Messages arrive.



**Figure 6-7: Simplified Processing Timing Requirements among DAP Processes**

**Table 6-7: DAR DB 808 Byte DAP User File (UserID File) TGBFS Message Record Format**

Length (Bytes)	Bits Req.	Format Type	Value Range	Bit Num. /Units	Description
4	32	Integer	0 - 1800	sec	2 second step offsets from hour
4	7	Integer	0-127		Existing Msg Indexing Flags (MsgMap)
					Bit Pattern Interpretation
				0	EMC Control Msg (0=No/1=Yes)
				1	Report Request Msg (0=No/1=Yes)
				2	Element Data Msg (0=No/1=Yes)
				3	Direction Cosines Msg (0=No/1=Yes)
				4	IBUG Control Msg (0=No/1=Yes)
				5	TRDS Array Geometry Msg (0=No/1=Yes)
				6	Fixed Weight Data Msg (0=No/1=Yes)
			7-30		Spare
	1			31	TGBFS Init Flag (0=No/1=Yes)
12					EMC Control (TGBFS 1Msg 0)
12					Report Request (TGBFS 1Msg 1)
248					Element Data (TGBFS 1Msg 2)
20					Direction Cosines (TGBFS 1Msg 3)
16					IBUG Control (TGBFS 1Msg 4)
244					TRDS Array Geometry (TGBFS 1Msg 5)
248					Fixed Weight Data (TGBFS 1Msg 6)

**Table 6-8: Process C Block Data Time Header Format**

Length (Bytes)	Bits Req.	Format Type	Value Range	Bit Num. /Units	Description
8	32	Unsigned Long Integer	0 - ( $2^{32}$ - 1)	sec	Send time (in Unix system time)
4	16	Unsigned Integer	0 - 65535		Number of bytes in all segments (TotalSize)
4	16	Unsigned Integer	0-350		Number of TGBFS messages (NumMsgs)

**Table 6-9: Temporary Directory File Record Fields**

Field Number	Field Name	Description
1	Directory Name	ACII String containing the directory name in YYYYDDHH format

**Table 6-10: USERFILENAME File Record Fields**

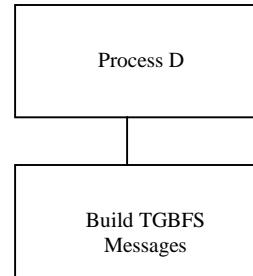
Field Number	Field Name	Description
1	File Name	ACII String containing the ID of the DAP DAR User

## 7. DAP Process D TGBFS Data Distribution Detailed Design

Process D is destined to be the highest priority in the system. This design covers the case of multiple DAP Users with a single instantiation of itself. If the process is forked (multiple instantiations) in the future the design will be modified. The following assumptions are made for Process D functions during Phase 1 implementation:

- Process D unpacks the segment groups from the data block produced by Process C
- Process D reads the Time Header on each segment group and sends it in real-time to the TGBFS Emulator
- Process D sends the segment groups at a 2 sec data rate without fail

Figure 7-1 shows the structure chart for functions of Process D during Phase 1 implementation.



**Figure 7-1: DAP Phase 1 Procedure Hierarchy for Process D**

## 7.1 PDL Procedures for Process D

The following subsections describe the PDL for Process D functions during Phase 1 implementation.

### 7.1.1 Begin Process D

```

Begin Process D
// TGBFS to DAP Interface Manager
    Initialize process
    Set wait time to forever
    Do while TRUE // endless main processing loop
        // give up CPU until new msg arrives or wait time expires
        Wait for new message 1000 from Process C
        If Process C message has arrived
            Get message 1000 from Process C
            // assess message type and take the appropriate action
            If message type 1000 // data block for TGBFS message construction (Fig 7-1)
                Store data block in InputBlockBuffer
                Flag new data block as valid
            End if
            If message type 1001 // current data block has been updated, so ignore [not in
                Phase 1]
                Flag current data block as invalid
            End if
        End if
        // extract data from block if the time has come to send it to the TGBFS
        If current message data block is valid // all will be valid for Phase 1
            Do
                Check next segment group header for time of transmission to TGBFS
                If current time is greater than or equal to block segment send time
                    Get segment group size and number of messages from the time header
                    Initialize pointers and counters for unpacking the block
                    Do
                        Perform Build TGBFS Messages
                    Until group segment send time exceeds current time
                    // give-up CPU to Processes A, B, C, E, F G
                    Set wait time to send time minus the current time
                End if
                Until entire InputBlockBuffer has been set
            Else
                Set wait time to forever // give-up CPU to Processes A, B, C, E, F G
            End if
        End do while // endless main processing loop
    End Process D

```

### 7.1.2 Procedure Build TGBFS Messages

Procedure Build TGBFS Messages

```
Do while SegByteCount <= TotalNumBytes
    Look up MsgSize based on Msg ID
    MsgByteCount = 0
    Set MsgBufferPointer = OutMsgBuffer Address
    Do while MsgByteCount < MsgSize
        SegByteCount = SegByteCount + 1
        MsgByteCount = MsgByteCount + 1
        OutMsgBuffer[MsgBufferPointer] = InputBlockBuffer[SegGroupPointer]
        BytePointer = BytePointer + 1
        SegGroupPointer = SegGroupPointer + 1
    End do while
    Send OutMsgBuffer to TGBFS Emulator
End do while
End Procedure Build TGBFS Messages
```

## 7.2 Process D Message Formats

### 7.2.1 Process D Input Messages

**Table 7-1: Process D Input Messages**

MSG ID	Message Name	Source	Description
1000	Data Block	Process C	Contiguous block of TGBFS message data for the next send time window
1001	Ignore Data Block (Beyond Phase 1)	Process C	Flag to ignore the current data block in Process D's memory

### 7.2.2 Process D Output Messages

**Table 7-2: Process D Output Messages**

MSG ID	Message Name	Destination	Description
0	EMC Control	TGBFS Emulator	See TGBFS ICD Section 5.3
1	Report Request	TGBFS Emulator	"
2	Element Data	TGBFS Emulator	"
3	Direction Cosines	TGBFS Emulator	"
4	IBUG Control	TGBFS Emulator	"
5	TDRS Array Geometry	TGBFS Emulator	"
6	Fixed Weight Data	TGBFS Emulator	"

### 7.2.3 Process D Data Inputs

Process D data inputs are the same as Process C Data Outputs.

### 7.2.4 Process D Data Outputs

Process D data outputs are described in the TGBFS ICD with regard to the messages referenced above.

## 8. Demand Access Buffer (DAB) Emulator Detailed Design

No assumptions are made for the DAB Emulator functions for Phase 1 implementation.

### 8.1 PDL Procedures for DAB Emulator

The following subsections describe the PDL for DAB Emulator functions during Phase 1 implementation.

#### 8.1.1 Process DAB Emulator

Process DAB Emulator

Do while TRUE // **endless loop**

    Wait for a message

    Get MsgID

    Case (MsgID)

##### 500: // MAF data handling service specification

        Extract service action from message

        If Add then

            Display "MAF Command and Data Buffering, Formatting, and Routing ADD Service Specifications Received from the DAP" in the DAB Emulator XTERM Window

            Display TDRS ID, USID, Start Time, Stop Time in the DAB Emulator XTERM Window

        Else

            Display " MAF Command and Data Buffering, Formatting, and Routing DELETE Service Specifications Received from the DAP " in the DAB Emulator XTERM Window

            Display TDRS ID, USID, Start Time, Stop Time in the DAB Emulator XTERM Window

        End if

##### 501: // DAR data handling service specification

        Extract service action from message

        If Add then

            Display "DAR Data Buffering, Formatting, and Routing ADD Service Specifications Received from the DAP" in the DAB Emulator XTERM Window

            Display TDRS ID, USID, Start Time, Stop Time in the DAB Emulator XTERM Window

        Else

Display " DAR Data Buffering, Formatting, and Routing DELETE Service  
Specifications Received from the DAP " in the DAB Emulator  
XTERM Window

Display TDRS ID, USID, Start Time, Stop Time in the DAB Emulator XTERM  
Window

End if

Other: // **erroneous msg received**

Display "DAR Data Buffering, Formatting, and Routing ADD Service Specifications  
Received from the DAP" in the DAB Emulator XTERM  
Window

End case

End do while

End Process DAB Emulator

## 9. Flight Dynamics Facility (FDF) Emulator Detailed Design

No assumptions are made for the FDF Emulator functions for Phase 1 implementation.

### 9.1 PDL Procedures for FDF Emulator

The following subsections describe the PDL for FDF Emulator functions during Phase 1 implementation.

#### 9.1.1 Process FDF Emulator

Process FDF Emulator

Do while TRUE // **endless loop**

Get the command line input CMD from keyboard

Case (CMD)

‘GenEphem’:

    Get Ephemeris Specification File Name from the Command Line

    Open Ephemeris Generation Specification File // **Table 9-1**

    Read ephemeris file name from Ephemeris Specification File

    Open Ephemeris File // **Table 9-2**

    Read ephemeris parameters start time, length, and time step

    Read the initial state vector

    Perform Propagate State Vector

    Close Ephemeris File

    Close Ephemeris Specification File

    Display ‘Ephemeris Generation Completed’ in FDF Emulator XTERM Window

‘SendEphem’:

    Read Ephemeris File Name from the Command Line

    Open Ephemeris File

    Do while (not EOF for Ephemeris File)

        Read record containing time and position vector components

        Format message 700

        Send Msg 700 to Process B

    End do while

    Close Ephemeris File

    Display ‘Ephemeris Sent to DAP’ in FDF Emulator XTERM Window

Other: // **command line error**

    Display ‘Unknown Command’ in FDF Emulator XTERM Window

    End case

    End do while

End FDF Emulator Process

### 9.1.2 Procedure Propagate State Vector

Procedure Propagate State Vector

```

Set tSTART ← start time
Set TEMPHEM ← ephemeris length
Set Δt ← time step
Perform Attitude Parameters α,β,γ
Write tSTART, x, y, z, α,β,γ to Ephemeris File
t ← tSTART
Do while t ≤ TEMPHEM
    t ← t + Δt
    Propagate last state vector to t // use J2 propagator (already developed in C)
    Perform Attitude Parameters α,β,γ
    Write t, x, y, z, α,β,γ to Ephemeris File
End do while
End Procedure Propagate State Vector

```

### 9.1.3 Procedure Attitude Parameters

Procedure Attitude Parameters

// Phase 1 simplification

```

α ← const
β ← const
γ ← const
End Procedure Attitude Parameters α,β,γ

```

**Table 9-1: Ephemeris Specification File Layout**

<b>Record No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>No. Bytes</b>	<b>Units</b>	<b>Description</b>
1	Ephemeris File	Char String	80 (Max)	N/A	Name of the ephemeris file to be generated
2	Start time	Double	8	Sec	Start time of the ephemeris to be generated
3	Length	Double	8	Sec	Length of the ephemeris
4	Time step	Double	8	Sec	Time separation between ephemeris entries

**Table 9-2: Ephemeris File Record Layout**

<b>Field No.</b>	<b>Field Name</b>	<b>Units</b>	<b>Data Type</b>	<b>No. Bytes</b>	<b>Description</b>
1	Time	Sec	Double	8	Time of the ephemeris entry
2	X	Km	Double	8	X coordinate
3	Y	Km	Double	8	Y coordinate
4	Z	Km	Double	8	Z coordinate
5	Alpha	Degrees	Double	8	Pitch
6	Beta	Degrees	Double	8	Roll
7	Gamma	Degrees	Double	8	Yaw

## 10. Network Communications Center (NCC) Emulator Detailed Design

No assumptions are made for the NCC Emulator functions for Phase 1 implementation.

### 10.1 PDL Procedures for NCC Emulator

The following subsections describe the PDL for NCC Emulator functions during Phase 1 implementation.

#### 10.1.1 NCC Message Handler Process

NCC Message Handler Process

Do while TRUE

    Wait for message from Process A // **endless loop**

    Get MsgID

    Case Msg ID

        400:

            Display ‘MAF Add Request Received from the DAP’ in NCC Emulator XTERM Window

            Format Msg 402

            Send Msg 402 to Process A

        401:

            Display ‘MAF Delete Request Received from the DAP’ in NCC Emulator XTERM Window

            Format Msg 403

            Send Msg 403 to Process A

        Other:

            Display ‘Message Error: Incorrect Message Received’ in NCC Emulator XTERM Window

            End case

        End do while

    End NCC Message Handler Process

### 10.1.2 NCC Emulator Operator Process

NCC Emulator Operator Process

Do while TRUE

    Get command input from keyboard

    Case (CMD)

        ‘GenTUT’:

            Get TUT Specification File Name from the Command Line

            Open TUT Generation Specification File // **Table 10-1**

            Read Input TUT File name from TUT Specification File

            Read TUT MAF start time

            Perform Translate MAF TUT

            Close TUT Specification File

            Display ‘TUT File Generation completed’ in FDF Emulator XTERM Window

        ‘SendTUT’:

            Read TUT File Name from the Command Line

            Open TUT File

            Do while (not EOF for TUT File)

                Read record containing time and position vector components

                Format message 600

                Send Msg 600 to Process B

            End do while

            Close TUT File

            Display ‘TUT Sent to DAP’ in NCC Emulator XTERM Window

        Other: // **command line error**

            Display ‘Unknown Command’ in NCC Emulator XTERM Window

    End case

End do while

End NCC Emulator Operator Process

### 10.1.3 Procedure Translate MAF TUT

Procedure Translate MAF TUT

    Open Input TUT File

    Open Output TUT File

    Do while (not EOF for Input TUT File)

        Read record from Input TUT File // **ASCII file from NCC**

        If MAF record // **modify MAF time scale**

            Translate existing time on Input TUT File record by  $\Delta t$  relative to the start time

            Insert translated time into Output TUT File record time field

            Copy remaining Input TUT File record fields to Output TUT File record

        End if

        Write record to Output TUT File

    End do while

    Close Output TUT File

    Close Input TUT File

End Perform Translate MAF TUT

**Table 10-1: TUT Specification File Layout**

<b>Record No.</b>	<b>Field Name</b>	<b>Data Type</b>	<b>No. Bytes</b>	<b>Units</b>	<b>Description</b>
1	TUT File	Char String	80 (Max)	N/A	Name of the TUT file to be generated
2	Start time	Double	8	Sec	MAF start time of the TUT to be generated

## 11. TGBFS Emulator Detailed Design

The following assumptions are made for the TGBFS Emulator functions during Phase 1 implementation:

- The emulator receives TGBFS messages from DAP Process D
- Each message received is analyzed to determine the type
- The message types are displayed in real-time scrolling fashion on the display
- Requests for TGBFS status are met with status reports to the DAP Status Reporting Process.

### 11.1 *PDL Procedures for TGBFS Emulator*

The following subsections describe the PDL for TGBFS Emulator functions during Phase 1 implementation.

### 11.1.1 Begin Process TGBFS Emulator

Begin Process TGBFS Emulator

    Initialize process

    Do while TRUE // **endless main processing loop**

        Wait until a TGBFS message is received from Process D

        Get message type

        Case TGBFS Message Type // **handle all possible inputs from the DAP**

            Msg Type 0:

                Display ‘EMC Control Message Type 0 Received from DAP’

            Msg Type 1:

                Display ‘Report Request Message Type 1 Received from DAP’

            Msg Type 2:

                Display ‘Element Data Message Type 2 Received from DAP’

            Msg Type 3:

                Display ‘Direction Cosines Message Type 3 Received from DAP’

            Msg Type 4:

                Display ‘IBUG Control Message Type 4 Received from DAP’

            Msg Type 5:

                Display ‘TDRS Array Geometry Message Type 5 Received from DAP’

            Msg Type 6:

                Display ‘Fixed Weight Data Message Type 6 Received from DAP’

        End case

    // **React to TGBFS status requests by sending a report to the DAP Process G**

    If Msg Type = 1 then

        Case TGBFS Report Type // **handle all possible outputs to DAP Process G**

            Report Type 0:

                Send ‘EMC Report Type 0 via Msg 10’

            Report Type 1:

                Send ‘EMC Extended Status Report Type 1 via Msg 11’

            Report Type 2:

                Send ‘IBUG Report Type 2 via Msg 12’

            Report Type 3:

                Send ‘IBUG Extended Status Type 3 via Msg 13’

            Report Type 4:

                Send ‘Calibration Report Type 4 via Msg 14’

            Report Type 5:

                Send ‘Noise Estimate Report Type 5 via Msg 15’

            Report Type 6:

                Send ‘Covariance Type 6 via Msg 16’

            Report Type 7:

                Send ‘Weights Type 7 via Msg 17’

        End case

    End if

End do while

End Process TGBFS Emulator

## 11.2 TGBFS Emulator Message Formats

### 11.2.1 TGBFS Emulator Input Messages

**Table 11-1: TGBFS Emulator Input Messages**

MSG ID	Message Name	Source	Description
0	EMC Control	DAP Process D	See TGBFS ICD Section 5.3
1	Report Request	DAP Process D	"
2	Element Data	DAP Process D	"
3	Direction Cosines	DAP Process D	"
4	IBUG Control	DAP Process D	"
5	TDRS Array Geometry	DAP Process D	"
6	Fixed Weight Data	DAP Process D	"

### 11.2.2 TGBFS Emulator Output Messages

**Table 11-2: TGBFS Emulator Output Messages**

MSG ID	Message Name	Destination	Description
10	EMC (Report 0)	DAP Process G	See TGBFS ICD Section 5.3
11	EMC Extended Status (Report 1)	DAP Process G	"
12	IBUG (Report 2)	DAP Process G	"
13	IBUG Extended Status (Report 3)	DAP Process G	"
14	Calibration Vector (Report 4)	DAP Process G	"
15	Noise Estimate (Report 5)	DAP Process G	"
16	Covariance (Report 6)	DAP Process G	"
17	Weights (Report 7)	DAP Process G	"

## 12. DAP Prototype Interface Message Numbering Scheme

**Table 12-1: DAP Prototype Interface Message Numbering Scheme**

Interface	Message Number Range
DAP Process D and TGBFS Emulator	0 - 99*
PT and DAP Process A	100 - 199
DAP Process A and DAP Process B	200 - 299
DAP Process A and DAP Process C	300 - 399
DAP Process A and NCC Emulator	400 - 499
DAP Process A and DAB Emulator	500 - 599
DAP Process B and NCC Emulator	600 - 699
DAP Process B and FDF Emulator	700 - 799
DAP Process G and TGBFS Emulator	800 - 899
DAP Process G and DAB Emulator	900 - 999
DAP Process C and DAP Process D	1000 - 1099
Spare	1100-1999

\* Special case header required since message structure is already defined

**Table 12-2: DAP Prototype Standard Header**

Item #	# Bytes	Data Type	Bit No.	Data Field	Description
1	2	Short integer (16 bits)	0-11	Message Number	See Table Above

Overall Message Number Range: 0 - 1999 ( $1999_{10} = 7CF_{16} = 11110101111_2$ )

## 13. Abbreviations and Acronyms

CPU	Central Processing Unit
DA	Demand Access
DB	Data Base
DAB	Demand Access Buffer
DAF	Demand Access Forward
DAP	Demand Access Processor
DAPP	Demand Access Processor Prototype
DAR	Demand Access Return
DAS	Demand Access System
DASDOC	Demand Access System Description and Operations Concept
DRS	Data Recovery Subsystem
FDF	Flight Dynamics Facility
GSFC	Goddard Space Flight Center
GT	Ground Terminal
GUI	Graphical User Input
IF	Intermediate Frequency
IP	Internet Protocols
IPC	Inter Process Communications
IUB	Independent Beamformer Unit
MA	Multiple Access
MAF	Multiple Access Forward
MAR	Multiple Access Return
msg	message
NCC	Network Control Center
NFS	Network File System
PN	Pseudorandom Noise
PT	Planning Tool
RF	Radio Frequency
SGL	Space Ground Link
SGLT	Space Ground Link Terminal
SHO	Schedule Request Order
SN	Space Network
STDN	Spaceflight Tracking and Data Network
SV	State Vector
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
TDM	Tracking Data Message
TGBFS	Third Generation Beamforming System
TCP	Telecommunications Protocol
TUT	TDRS Unused Time
UNIX	SUN Operating System
UP	User Platform
USAT	User Satellite
WSC	White Sands Complex

## 14. DAP PDL Procedure Index

Add MAF Service.....	94, 95	Generate MAF User Availability Schedule	67, 69,
Add User Request.....	61, 72, 73, 74	85	
Assess Message .....	56, 57, 92, 93	Generate TDRS-USAT Visibility Schedule	61,
Assess Msg 200 .....	92, 94	62, 68, 87	
ATR-process.....	40	Get Ephem .....	92, 98
Attitude Parameters .....	127	Get TUT.....	92, 98, 130
Availability Intervals .....	75, 79, 80	Get UserID File Name .....	106
Available Time Message ( From DAP to PT) (Msg 111) .....	46	Implement MAF Service Request.....	56, 61
Available Time Request Message (From PT to DAP) (Msg 101).....	43	Initialize DARDB Structure.....	62, 64
Begin Process A.....	56	Insert Msgs into UserID File.....	63, 66, 67
Begin Process B.....	92	Insert TGBFS Messages into Block.....	106, 107
Begin Process C.....	104	Load DARDB .....	62, 63, 86
Begin Process D.....	121	MAF Delete Arrival.....	56
Begin Process TGBFS Emulator .....	134	MAF Planning Information.....	67
Build Message 111 .....	75	MAF Schedule Delete Request .....	56, 58
Build Message 112 .....	75	MAF service request.....	16, 18, 56, 61
Build Message 113 .....	76	Make Group Segment Time Header .....	106, 107
Build Message 200 .....	76	Modify TUT .....	94
Build Message 300 .....	77	NCC Emulator Operator Process .....	130
Build Message 400 .....	77	NCC Message Handler Process .....	129
Build Message 401 .....	77	PDL Procedures for DAB Emulator .....	124
Build Message 500 .....	78	PDL Procedures for FDF Emulator .....	126
Build Message 501 .....	78	PDL Procedures for NCC Emulator.....	129
Build Msg Buffer.....	63, 65	PDL Procedures for Planning Tool.....	37
Build Segments.....	106	PDL Procedures for Process A .....	56
Build TGBFS Messages .....	62, 121, 122	PDL Procedures for Process B.....	92
Calculate TDRS-USAT Visibility .	68, 69, 70, 71	PDL Procedures for Process C.....	104
Conflict Check.....	73, 74	PDL Procedures for Process D .....	121
Create Entirely New TUT.....	96, 97	PDL Procedures for TGBFS Emulator .....	133
Create Interp EPH File .....	87, 98, 99, 100	Preliminary Message Interface Definitions	
DAPMsg_process .....	41	Between DAP and PT .....	27
DAPThread.....	39	Procedure Build_Block.....	105
DAR Planning Information.....	67	Process A Data Inputs.....	84
DAR return request.....	18, 56, 62	Process A Data Outputs .....	83
DAR Schedule Delete Request .....	56, 59	Process A Input Messages .....	82
Delete MAF Service .....	94, 96	Process A Message Formats .....	81
Delete TGBFS Messages.....	59, 60, 86, 87	Process A Output Messages .....	82
Delete User Request .....	72, 74	Process A, B, and C Shared Files	
Determine Directories.....	59, 60, 86, 105	Process A .....	85
Determine Required TGBFS Msgs.....	63, 64	Process B Data Inputs .....	100
Enlarge TUT at End.....	96, 97	Process B Input Messages .....	100
Enlarge TUT at Start.....	96, 97	Process B Message Formats .....	100
Extract Msgs .....	105, 106	Process C Data Inputs .....	113
Forward Service Data Message ( From DAP to PT) (Msg 114) .....	49	Process C Data Outputs .....	114, 123
Gather TGBFS Msg Data .....	105	Process C Input Messages .....	113
Generate DAR User Availability Schedule	67, 70,	Process C Message Formats .....	113
85, 87		Process C Output Messages .....	113

Process D Output Messages.....	123	Schedule Result Message ( From DAP to PT)	
Process FDF Emulator.....	126	(Msg112) .....	47
Propagate State Vector .....	126, 127	SDR-process .....	40
PT_Main_GUI.....	38	Start or Stop Getting Ephem.....	92, 98
PTNetwork .....	38, 39	TGBFS Emulator Input Messages .....	135
Return Service Data Message Format ( From DAP to PT) (Msg 115) .....	50	TGBFS Emulator Message Formats .....	135
SAT-process .....	40	TGBFS Emulator Output Messages.....	135
Schedule Add Request Message ( From PT to DAP) (Msg 102) .....	44	Timing Requirements Among DAP Processes	115
Schedule Delete Request Message ( From PT to DAP) (Msg 103) .....	45	Translate MAF TUT .....	130, 131
Schedule Deletion Notification Message ( From DAP to PT) (Msg 113) .....	48	Update DAR User Schedule .....	59, 62, 72
		Verify DAR Service Exists .....	59
		Verify MAF Service Exists .....	58, 85